Bayesian Data Analysis

Practical Data Analysis with Bugs using R

Computer Exercises

Final draft , 10th August 2008
www.pubhealth.ku.dk/~bxc/Bayes/Cph-2008

Copenhagen

August 2008

Lyle Gurrin School of Population Health
University of Melbourne
lgurrin@unimelb.edu.au
http://www.epi.unimelb.edu.au/about/staff/gurrin-lyle.html

Bendix Carstensen Steno Diabetes Center, Gentofte
Department of Biostatistics, University of Copenhagen
bxc@steno.dk
http://www.biostat.ku.dk/~bxc

Søren Højsgaard Bioinformatics, Genetics and Statistics Research Unit
Institute of Genetics and Biotechnology, Aarhus University
soren.hojsgaard@agrsci.dk
http://genetics.agrsci.dk/~sorenh

Contents

Course program

Venue: CSS 2.1.02, Øster Farimagsgade 5:
http://maps.google.dk/maps?f=s&ie=UTF8&ll=55.687166,12.570956&spn=0.005927,0.021801&z=16
If you are in front of the big yellow-brick building with one gate on either side of the spire, choose the *right* gate, take the stairs *inside* the gate to the *right*, find the staircase and walk up one floor. You are now in building 2, 1st floor, i.e. 2.1. Then go to room 02.

Monday 11 August 2008

09:00 – 09:30	Registration & coffee.
09:30 – 10:15	**Lecture 1**: Introduction to Bayesian analysis: The binomial model as an example. (LG)
10:15 – 10:30	**Lecture 0**: Getting R and BUGS running. (BxC)
10:30 – 11:00	**Morning Tea**
11:00 – 12:30	**Practical 1**: Bayesian analysis in R: Discrete prior distribution in the DRUGS example. Illustration of posterior = likelihood × prior. The effect of data and prior variance using Beta probability functions in R. (BxC)
12:30 – 13:30	**Lunch**
13:30 – 14:30	**Lecture 2**: Introduction to MCMC and the BUGS programming language. (BxC/SH)
14:30 – 16:00	**Practical 2**: Simple analyses in BUGS using the binomial distribution, example of restricted uniform or beta prior distribution with narrow prior support for a range of parameters values. (BxC/SH)

Tuesday 12 August 2008

09:00 – 09:30	**Recap of Monday**
09:30 – 10:00	**Lecture 3**: Demonstrating the Gibbs sampler with a multiparameter problem and some data. The role of DAG-able models for the BUGS machinery to work. (SH)
10:00 – 10:30	**Practical 3**: The Gibbs sampler and the Metropolis-Hastings sampler with a bivariate normal example. (SH)
10:30 – 11:00	**Morning Tea**
11:00 – 11:30	**Lecture 4**: Poisson model for count data and rates. (LG)
11:30 – 12:30	**Practical 4**: Estimating the rate and time trend of asthma deaths in Australia using a Poisson model. (LG)
12:30 – 13:30	**Lunch**
13:30 – 14:00	**Lecture 5**: The normal model, multiparameter problems and the conceptually simple Bayesian approach. (LG)
14:00 – 14:45	**Practical 5**: Speed of light example showing the use of posterior predictive checking. First introduce a noninformative prior distribution for the mean and then an informative distribution - does this influence our opinion as to whether the lowest observations are outliers? (LG)
14:45 – 15:15	**Lecture 6**: Multiparameter generalized linear models. (LG)
15:15 – 16:30	**Practical 6**: Airline fatalities and posterior prediction of future fatalities: Several models: 1) Linear in log rate, 2) Linear in rate (problems with prior spec.) [1&2 simple in R.]. 3) Some parametric model of rate decay. (BxC/LG)

Wednesday 13 August 2008

09:00 – 09:30	**Recap of Tuesday**
09:30 – 10:00	**Lecture 7**: Monitoring convergence and the need to run multiple chains. (LG)
10:00 – 10:30	**Practical 7**: Problems with convergence - an example? (SH/LG)
10:30 – 11:00	**Morning Tea**
11:00 – 11:30	**Lecture 8**: Hierarchical models. (LG)
11:30 – 13:00	**Practical 8**: Meta-analysis of clinical trials as an example of a hierarchical model. (LG)
13:00 –	Afternoon free!

Thursday 14 August 2008

09:00 – 09:30	**Recap of Wednesday**
09:30 – 10:00	**Lecture 9**: Fetal growth example of linear mixed model. (LG)
10:00 – 10:30	**Morning Tea**
10:30 – 12:30	**Practical 9**: Fitting linear mixed models in R and BUGS: Fetal growth (head circumference) as a quadratic mean, random linear function of gestational age. Compare with SAS/R/Stata approach. Reporting essential. (LG)
12:30 – 13:30	**Lunch**
13:30 – 14:15	**Lecture 10**: Generalised linear mixed models (GLMMs) in BUGS. (LG)
14:15 – 16:00	**Practical 10**: Illustration of GLMMs using clustered binary data from GPs, also twin and family data with genetically structured covariance. (LG)
18:00 – 22:00	Course dinner.

Friday 15 August 2008

09:15 – 09:30	**Recap of Thursday**
09:45 – 10:30	**Lecture 11**: Model comparison using DIC. (LG)
10:30 – 11:00	**Morning Tea**
11:00 – 12:30	**Practical 11**: Comparing models in BUGS using DIC. (LG)
12:30 – 13:30	**Lunch**
13:30 – 14:15	**Lecture 12**: Comparing methods of measurement in Stata, SAS, GenStat, R and BUGS. (BxC)
14:15 – 15:30	**Practical 12**: Comparing methods of measurement using the MethComp package — reporting (BxC)
15:30 – 16:00	Wrapping up, closure, evaluation and farewell

Chapter 1

Introduction to computing and practicals

The course is both theoretical and practical, i.e. the aim is to convey a basic understanding of the Bayesian framework for data analysis as well practical computing skills in Bayesian methods. The two components of the course are supposed to support each other.

The practicals during the week will take place in computer labs, but the most convenient will be if you work on your own laptop for the practicals. This will ensure that useful scripts and trics are readily available for your future exploitation.

In the following is a brief overview of the software and other files you must download if you want to use your own computer.

1.1 Software

1.1.1 Overview

In this course, we use the Markov Chain Monte Carlo (MCMC) machinery which is implemented in various guises of `BUGS`. The original purpose of the software `BUGS` was to use it for Bayesian inference, but in many practical circumstances it is used with flat or (almost) uninformative prior distributions to effectively perform maximum likelihood inference.

The latter type of application is the main content of this course. But this use of the sotware still requires a basic knowledge of Baesian statistics.

The data manipulation and report generation is done with R in this course, as this is the state of the art in practical statistics. In order to avoid direct interaction with the `BUGS` programs, this course will use the `R2WinBUGS` interface, which basically throws R datastructures at the `BUGS` program and sucks the results back into R. This enables you to maintain a completely reproducible record of your initial data-manipulation (in R), estimation (in `BUGS`) and reporting of results (in R).

There are two version of `BUGS` we shall be using — you can choose which one suits you better — `WinBUGS` or `OpenBugs`. The scritping language is the same for the two, but `WinBUGS` is a separate program that is fired up and closed down from within R, whereas `OpenBugs` comes as an R-package, `BRugs`, that is operated entirely inside R.

In order to be able to write scripts (programs) in R and keep them for future use (and modification for other purposes) a good editor with interface to R is convenient. `Tinn-R` is the answer. (Tinn = Tinn Is Not Notepad). If you are already a user of ESS, just forget about `Tinn-R`.

So you need R, `BUGS` and (possibly) `Tinn-R`.

1.1.2 What to get

- Tinn-R is available from http://sourceforge.net/projects/tinn-r.
- R, version 2.7.1, get it from http://mirrors.dotsrc.org/cran/. The relevant packages for this course are easiest installed by fireing up R, and then type:

  ```
  > install.packages("R2WinBUGS","Brugs","coda","Epi")
  ```

  You will be asked to select a mirror (i.e. a computer) from which to download the stuff). The `R2WinBUGS` is the package that handles the interface to `BUGS`, `BRugs` is the `OpenBugs` program encapsulated in an R-package, `coda` is a package for post-processing and monitoring of MCMC-output, and `Epi` is a package for epidemiology from which we will use a few handy functions.
- `WinBUGS` from http://www.mrc-bsu.cam.ac.uk/bugs/.

  (If you have set your mind on using `OepnBugs` from `BRugs`, you can skip this section).

  You should get the update to 1.4.3, *and* a licence key for unrestricted use of `WinBUGS`. The licence key is free and will be sent to you by e-mail, or you can get it here: http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/WinBUGS14_key_31_12_2008.txt. It is just a plin text file that you have to paste into a certain window in `WinBUGS`. Without this licence key you cannot use `WinBUGS` for the practicals in this course, as you will experience limitations on the size of problems you can handle..

1.1.3 How to install and fine-tune

1.1.3.1 Tinn-R and R

R can run in two different ways on your computer: MDI or SDI. MDI is "multiple display interface", where the command window, graph, R-editor and help windows all are sub-windows in a master-window. This is the default and is *not* supported from `Tinn-R`. In order to get R to start in SDI ("single diaplay interface") mode where each window is stand-alone you must edit the `Rconsole` file that is located in the folder `c:/Program Files/R-2.7.1/etc`. It is pretty self-explanatory what is in that file; so you must put `MDI=no`. You may also wish to change the colors of the command screen to less eye-straining colors, e.g.:

```
> background = gray7
> normaltext = yellow2
> usertext = green
> highlight = white
```

You can also change the default font size by editing this file.

1.2 Course material

Datsets and programs for the course are all collected in the zip file `BDA2008.zip` available at the course homepage, www.biostat.ku.dk/~bxc/Bayes/Cph-2008/. Download this file and unpack it in a separate folder. The resulting folder tree has the following subfolders:

- Data — datasets for use in the practicals.
- R — example R-programs providing solutions to some of the practicals, as well as the file `PDAwBuR.r` which contains a couple of ad-hoc R-functions that should be handy in some of the exercises.

At the root level you should find this document, including solutions to the exercises.

1.3 Simulating data in R

One of the major uses of computers in this course is simulation, a brief section on this is include here.

Start by opening R. In the following, "`>`" is the R-prompt, and "`+`" the continuation prompt, and these should not be typed. The lines starting with "`[1]`", "`[8]`" etc. are output from R, that you can use to check that you got the right output. Since this is about simulation, you will of course not get exactly the same output as shown here.

To simulate binomial variates $Y \sim \text{Bin}(N, p)$, the function to use is `rbinom`. To simulate $n = 1$ observation from one experiment of size $N = 10$ and a probability of success $p = 0.2$, try the following:

```
> rbinom(n=1,size=10,prob=0.2)

[1] 2
```

In many cases we want to make such simulations several times. To conduct the experiment, say 15 times we can do:

```
> rbinom(n=15,size=10,prob=.2)

 [1] 2 2 3 4 2 2 2 0 4 0 1 2 3 2 3
```

Sampling from a Bernoulli distribution (which is just a $\text{Bin}(1, p)$–distribution) is therefore achieved by

```
> rbinom(n=15,size=1,prob=.2)

 [1] 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0
```

or simply

```
> rbinom(15,1,.2)

 [1] 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0
```

For more information on `rbinom` type `?rbinom`. Similarly, random normal and Poisson variates are generated using `rnorm` and `rpois`. For information on these, type `?rnorm` or `?rpois`.

If you want to take a random sample from the elements of a vector you need the function `sample`. First look at the vector from 1 to 10:

```
> 1:10

 [1]  1  2  3  4  5  6  7  8  9 10

> sample( 1:10, 8, replace=T )

[1] 7 7 4 2 7 5 6 4
```

Here we took a sample of 8 from the vector $(1, 2, \ldots, 10)$, with replacement. If you want a sample without replacement, just do:

```
> sample( 1:10, 8 )

[1]  9  8  6  4  5 10  3  7
```

If you omit the second argument, you just get a permutation of the argument:

```
> sample( 1:10 )

 [1]  3  7  2  5  8  6  1  4 10  9

> sample( 1:10 )

 [1]  7  4  6  5  8 10  1  3  2  9
```

1.4 Distributions in R

All the standard distributions are avialble in R; for example the normal distribution density is called by `dnorm`, the cumulative distribution is called `pnorm`, the invers of this `qnorm`, and a random sample from it generted by `rnorm`.

In general any distribution has these four functions associated with it.

There is a function in the `MASS` library (which is by defualt included in any R-installation) to generate random samples from a multivariate normal distributioon, `mvrnorm`.

1.5 Using the interface to BUGS

This brief "Practice 0" is to get you familiar with the practicalities around running `BUGS` from within R and making sure that the installation on your computer works. It is not a proper exercise but meant for use as a check of your computing installation.

We are going to analyze the annual number of airline fatalities using a simple Poisson model and use this model to predict the future number of fatalities. This corresponds to the first part of exercise 6.

First get the data and take a look at it:

```
> airline <- read.csv( "../data/airline.csv" )
> airline
```

```
   year1975 year fatal  miles  rate
1         1 1976    24  3.863 6.213
2         2 1977    25  4.300 5.814
3         3 1978    31  5.027 6.167
4         4 1979    31  5.481 5.656
5         5 1980    22  5.814 3.784
6         6 1981    21  6.033 3.481
7         7 1982    26  5.877 4.424
8         8 1983    20  6.223 3.214
9         9 1984    16  7.433 2.152
10       10 1985    22  7.107 3.096
11       11 1986    22  9.100 2.418
12       12 1987    25 10.000 2.500
13       13 1988    29 10.600 2.736
14       14 1989    29 10.988 2.639
15       15 1990    27 10.880 2.482
16       16 1991    29 10.633 2.727
17       17 1992    28 11.956 2.342
18       18 1993    33 12.343 2.674
19       19 1994    27 13.011 2.075
20       20 1995    25 14.220 1.758
21       21 1996    24 16.371 1.466
22       22 1997    26 15.483 1.679
23       23 1998    20 18.080 1.106
24       24 1999    21 16.633 1.263
25       25 2000    18 18.875 0.954
26       26 2001    13 19.233 0.676
```

We shall only be interested in the column `fatal` which contains the annual number of fatalities. We use the following model to describe the number of fatalities in year i, y_i:

$$y_i|\mu \sim \text{Poisson}(\mu), \quad \mu \sim \Gamma(0,0)$$

The $\Gamma(0,0)$ is really the uniform distribution on $(0,+\infty)$, (so an improper prior), but it will work, as the posterior for μ will be $\Gamma(0+\sum y_i, 0+n)$ where n is the number of observations, in this case 26, and $\sum y_i = 634$.

Since we know the posterior distribution, we can compute the mean and median of this by simulating a sample of say 1000 from it:

```
> ( mn <- mean( xx <- rgamma( 10000, 634, 26 ) ) )

[1] 24.38316

> ( md <- median( xx ) )

[1] 24.36896
```

We can also draw the posterior distribution for μ, with indication of the mean and median:

```
> curve( dgamma( x, 634, 26 ), from=20, to=30, lwd=4 )
> abline( v=mn, col="red" )
> abline( v=md, col="blue" )
```

1.5.1 Using BUGS via bugs()

In order to run BUGS we must of course supply the data, but also a BUGS program as well as a couple of other things.

Data The first thing to provide to BUGS is the data. This is also provided in the form of a named list, one element per data-structure (usually vector or matrix). In this case we provide the vector of fatal airline accidents expanded with a NA for prediction of the number in 2002, as well as the total number of observations:

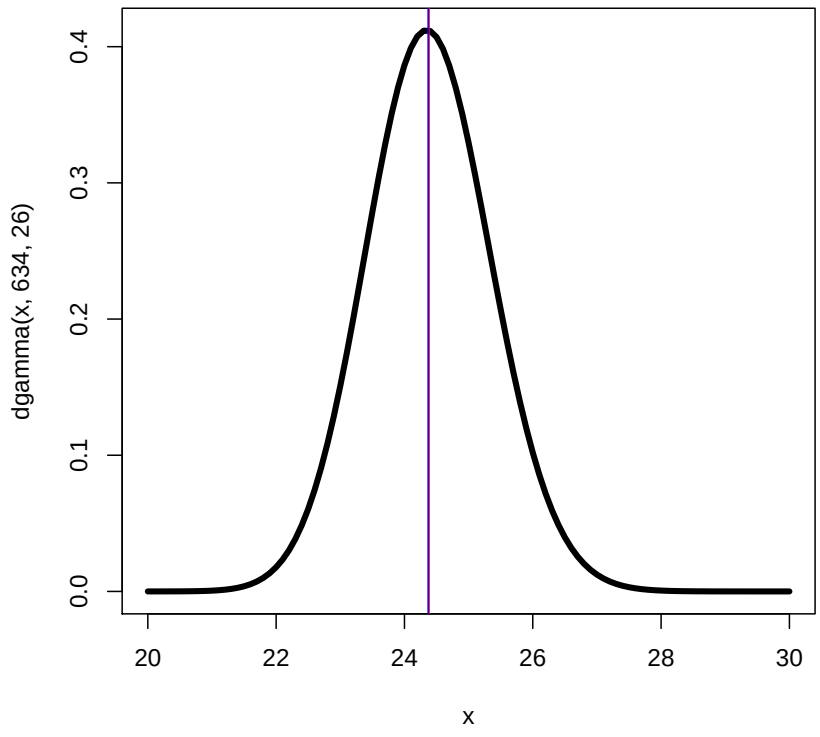

Figure 1.1: *The posterior distribution for mu. Mean is the red line, median the blue.*

```
> a.dat <- list( fatal = c(airline$fatal,NA), I=27 )
```

Program The program (BUGS code) must be put in a separate file which is then read by BUGS. When working in R this is most conveniently done using the R-function `cat()` which behaves pretty much like `paste()` with the exception that the result is written to a file you specify. If you specify `file=""` the output comes on your screen.

Here is the BUGS code specifying the above model, using `cat` to put it in the file `m1.bug`:

```
> cat( "model
+       {
+       for( i in 1:I )
+         {
+         fatal[i] ~ dpois(mu)
+         }
+       mu ~ dgamma(0,0)
+       }",
+      file="m1.bug" )
```

The code refers to data points in the variable `fatal` which is `I` long. The BUGS code is *declarative*, i.e. it is not executed as the program runs. Instead it is a specification of the model structure, and *after* the model is set up it is decided how best to go about the MCMC-simulation. So it would not matter if the specification of a prior ofr `mu` was put befor the `for` statement. Also the loop is just a compact way of writing `fatal[1] dpois(mu)`, `fatal[2] dpois(mu)`, `fatal[3] dpois(mu)` etc.

We could have replaced `I` with the number 27 in the code if we wanted. In that case the `I` in the data would have been superfluous, and you would get an error if you supplied the variable `I` — `OpenBugs` may even cause your entire R-session to exit without further ado if you supply variables in the data not used in the program.

Starting values To start the MCMC simulation we will normally supply some starting values (but in most cases BUGS will be able to generate them). In order to be able to monitor convergence we will normally run several chains, so we must supply starting values for each chain. The starting values for one chain is a named list, names are the parameters used in the model. Here we use three chains, hence the initial values is a list of three lists. Each of these list has as elements one named value for each parameter — in this case there is only one parameter μ, called `mu` in the BUGS program:

```
> a.ini <- list( list( mu=22 ),
+                list( mu=23 ),
+                list( mu=24 ) )
```

Parameters to monitor We must also specify the variables (nodes) that we want to monitor, this is done using the argument `parameters.to.save` (which can be abbreviated to `param`).

Simulation parameters Running the MCMC simulation via `bugs` also requires that we specify the (total) number of simulations (`n.iter`), the number of burn-in iterations (`n.burnin`), and the frequency of sampling for the simulations after the burn-in (`n.thin`). In the following example we run the chain for 3000 iterations, and use the first 2000 as burn-in and then sample every fifth, giving us a sample of 200 values from each if the 3 chains.

1.5.2 Results

Results from a MCMC sampling is a random sample from the joint posterior distribution of the parameters ("nodes") that have been monitored during running of the chain(s). This will normally

be represented in a matrix with one column for each parameter and one row for each sample, represented in the `coda` package as a `mcmc` object. This is a matrix with a bit of extra structure, but primarily there are a number of meningful functions associated with it ("`method`"s), notably `summary` and `plot`.

When we run more than one chain (which is the recommended approach) we wil have a number of such objects. These are represented in the `coda` package as `mcmc.list` objects. A `mcmc.list` object is basically just a list of `mcmc` objects. This also has a set of useful methods associated, such as `summary`, `plot`, `varnames`, and a range of plotting functions such as `xyplot`, `densityplot` and `acfplot` designed to monitor convergence of chains.

Because of the wide selection of methods for `mcmc.list` objects we encourage you to convert all results from MCMC simulations to `mcmc.list` objects.

Now we can run `BUGS` in three different ways:

- Using `WinBUGS` returning a `bugs` object.
- Using `WinBUGS` returning a text-string giving the names, enabling other programs to read the posteriors from the generated files.
- Using `BRugs` returning a `bugs` object.

For the two first options it is necessary to specify the path to the installation library for `WinBUGS`.

Also note that we enclosed all calls in a `system.time()` command to see how long each of them takes; it is only the last of the three numbers that is relevant, because R is not monitoring the time that `WinBUGS` is using.

```
> # Winbugs installation directory
> bd <- "c:/stat/bugs/winbugs14"
> # Using WinBUGS creating a bugs object
> system.time(
+ m1.wb <-
+ bugs( data = a.dat,
+      inits = a.ini,
+      param = c("mu","fatal[27]"),
+      model = "m1.bug",
+   n.chains = 3,
+     n.iter = 3000,
+   n.burnin = 2000,
+     n.thin = 5,
+ bugs.directory = bd,
+      debug = FALSE,
+    clearWD = TRUE ) )

   user  system elapsed
   0.08    0.00    6.19

> class( m1.wb )

[1] "bugs"

> # Using WinBUGS and converting to coda-format using codaPkg = TRUE
> system.time(
+ m1.coda <-
+ bugs( data = a.dat,
+      inits = a.ini,
+      param = c("mu","fatal[27]"),
+      model = "m1.bug",
+   n.chains = 3,
+     n.iter = 3000,
```

```
+   n.burnin = 2000,
+     n.thin = 5,
+ bugs.directory = bd,
+    codaPkg = TRUE,
+      debug = FALSE ) )

   user  system elapsed
   0.02    0.01    2.32

> class( m1.coda )

[1] "character"

> # Using OpenBUGS via the BRugs-package: bugs.directory is superfluous
> system.time(
+ m1.brugs <-
+ bugs( data = a.dat,
+      inits = a.ini,
+      param = c("mu","fatal[27]"),
+      model = "m1.bug",
+   n.chains = 3,
+     n.iter = 3000,
+   n.burnin = 2000,
+     n.thin = 5,
+    program = "openbugs",
+      debug = FALSE,
+    clearWD = TRUE ) )

Initializing chain 1: Initializing chain 2: Initializing chain 3:    user  system elapsed
   0.33    0.04    0.94

> class( m1.brugs )

[1] "bugs"
```

Currently there is no uniform way of converting a `bugs` object to a `mcmc.list` object, but in the course folder (in the file `PDAwBuR.r`) is a function that reads any kind of output from `bugs` into a `mcmc.list` object:

```
> source("../r/PDAwBuR.r")
> mcmc.list.bugs

function( x, ... )
{
  if (!is.R() && !require("coda"))
      stop("package 'coda' is required to use this function")
  if( is.character(x) )
     res <- mcmc.list(lapply(x,
                     read.coda,
                    index.file = file.path(dirname(x[1]),
                                           "codaIndex.txt"), ...))
  if( inherits(x,"bugs") )
    {
    zz <- list(list())
    aa <- x$sims.array
    for( i in 1:(dim(aa)[2]) )
       {
       tmp <- mcmc( aa[,i,] )
       zz <- c( zz, list(tmp) )
       }
    res <- mcmc.list( zz[-1] )
    }
  res
}
```

```
> # Convert them all to mcmc.list objects:
> mc.wb    <- mcmc.list.bugs( m1.wb )
> mc.coda  <- mcmc.list.bugs( m1.coda )

Abstracting deviance ... 200 valid values
Abstracting fatal[27] ... 200 valid values
Abstracting mu ... 200 valid values
Abstracting deviance ... 200 valid values
Abstracting fatal[27] ... 200 valid values
Abstracting mu ... 200 valid values
Abstracting deviance ... 200 valid values
Abstracting fatal[27] ... 200 valid values
Abstracting mu ... 200 valid values

> mc.brugs <- mcmc.list.bugs( m1.brugs )
> str( mc.brugs )

List of 3
 $ : mcmc [1:200, 1:3] 23.7 25.1 23.7 23.3 26.3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "mu" "deviance" "fatal[27]"
  ..- attr(*, "mcpar")= num [1:3] 1 200 1
 $ : mcmc [1:200, 1:3] 23.6 23.6 24.3 25.5 24.6 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "mu" "deviance" "fatal[27]"
  ..- attr(*, "mcpar")= num [1:3] 1 200 1
 $ : mcmc [1:200, 1:3] 25.3 24.8 26.4 23.7 23.3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "mu" "deviance" "fatal[27]"
  ..- attr(*, "mcpar")= num [1:3] 1 200 1
 - attr(*, "class")= chr "mcmc.list"
```

Once the objects are converted into `mcmc.list` objects you have access to a number of tools for summarizing the results and checking the convergence of the chains.

As always in R, there is a `summary` function:

```
> summary(mc.brugs)

Iterations = 1:200
Thinning interval = 1
Number of chains = 3
Sample size per chain = 200

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean    SD Naive SE Time-series SE
mu         24.40 0.970   0.0396        0.03939
deviance  156.23 1.445   0.0590        0.05719
fatal[27]  24.68 5.178   0.2114        0.20937

2. Quantiles for each variable:

            2.5%    25%    50%    75%  97.5%
mu         22.64  23.78  24.37  24.98  26.41
deviance  155.24 155.32 155.62 156.58 160.70
fatal[27]  15.00  21.00  25.00  28.00  35.02
```

It is possibile to explore the traces of the chains by the function `xyplot` (if you want to access the help-page use `?xyplot.mcmc`; `?xyplot` will give you the help-page for the basic `lattice` function):

```
> print( xyplot( mc.wb, main="WinBUGS direct" ) )
```

These plots use the `lattice` machinery for generating plots; it is only in interactive mode you can use the functions alone. If you want output on a file you must `print` them in order to get the plots onto a file, hence the `print()` surrounding the call of `xyplot` above.

```
> print( xyplot( mc.coda, main="WinBUGS via coda" ) )
```

A closer look at the traces of these two simulations will reveal that although run by two different simulations of `WinBUGS` they are identical. If you want a different starting point for the simulation you must supply a seed via the `bugs.seed`.

```
> print( xyplot( mc.brugs, main="BRugs" ) )
```

Clearly, all three trace plots look fine, so we conclude that the chain mixing is acceptable and we just proceed using the results from the `BRugs` run.

You can explore the posterior densities from each of the three chains by the command `densityplot`:

```
> print( densityplot( mc.brugs ) )
```

Once satisfied with convergence you can look at the posterior across all chains, by assembling them in an `mcmc` object. A few bells and whisthles have been added here, such as the omission of the density of the deviance (`[,-2]`), the constraining of the scales to be the same on the x-axis (`scales=list(x="same",y="free")`), the use of the entire plot area (`aspect="fill"`), and the arrangement of panels in 1 column by two rows (`layout=c(1,2)`):

```
> print(
+ densityplot( as.mcmc(as.matrix(mc.brugs))[,-2],
+              main="BRugs", lwd=3,
+              aspect="fill", scales=list(x="same",y="free"),
+              plot.points=FALSE, layout=c(1,2) ) )
```

If you want the density of one specific parameter only, you can go back to basics and use `density`. This also gives you the possibility of subsequently plotting the analytically derived density in this frame too:

```
> plot( density(mu.post<-as.mcmc(as.matrix(mc.brugs))[,"mu"]),
+       xlab=expression(mu), ylab="", lwd=4, col=gray(0.5), main="")
> abline(v=quantile(mu.post,probs=c(5,50,95)/100))
> curve( dgamma( x, 634, 26 ), from=20, to=30, lwd=3, col="red", add=TRUE )
```

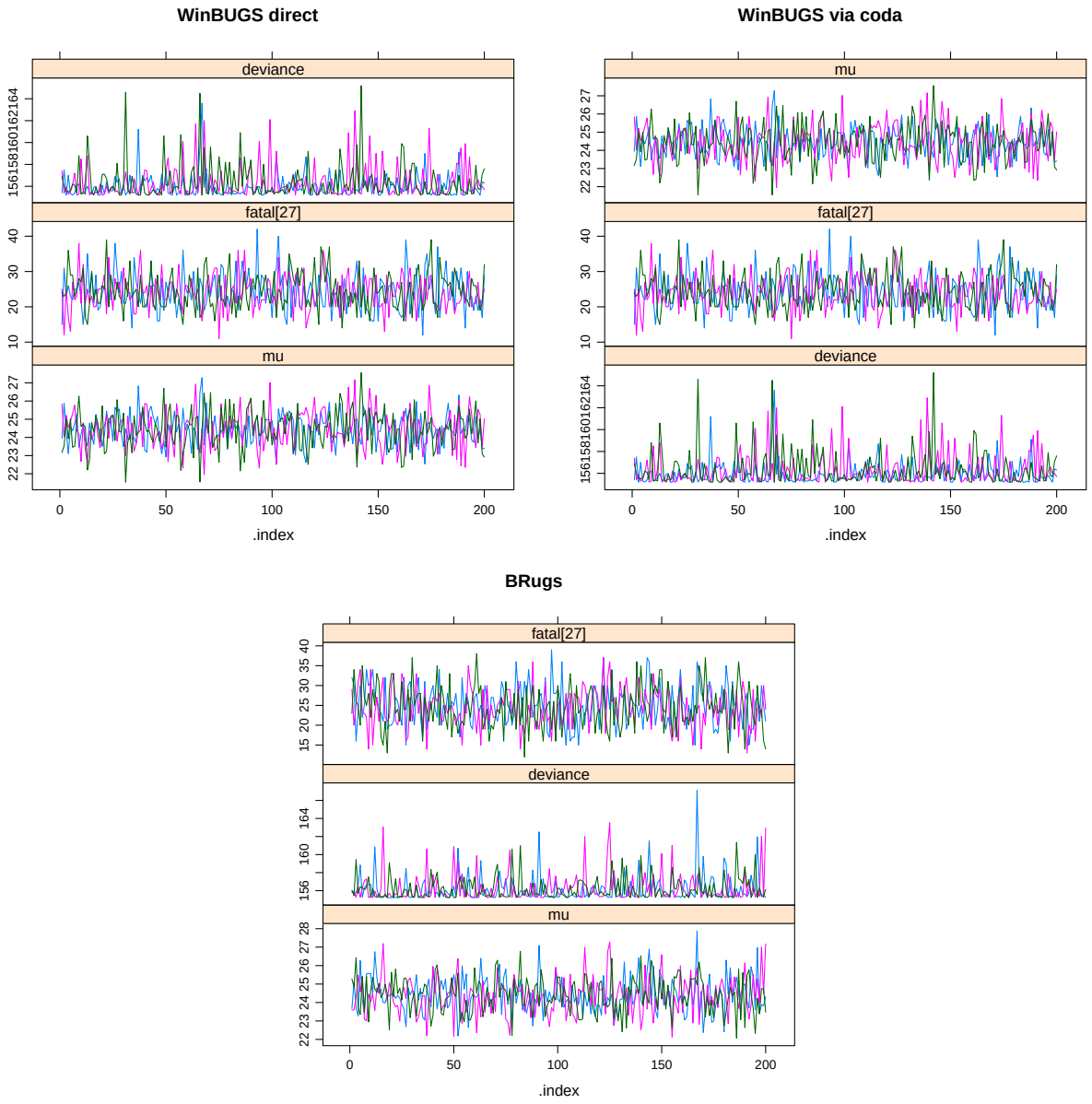

Figure 1.2: *Trace plots from the three different approaches to running* BUGS.

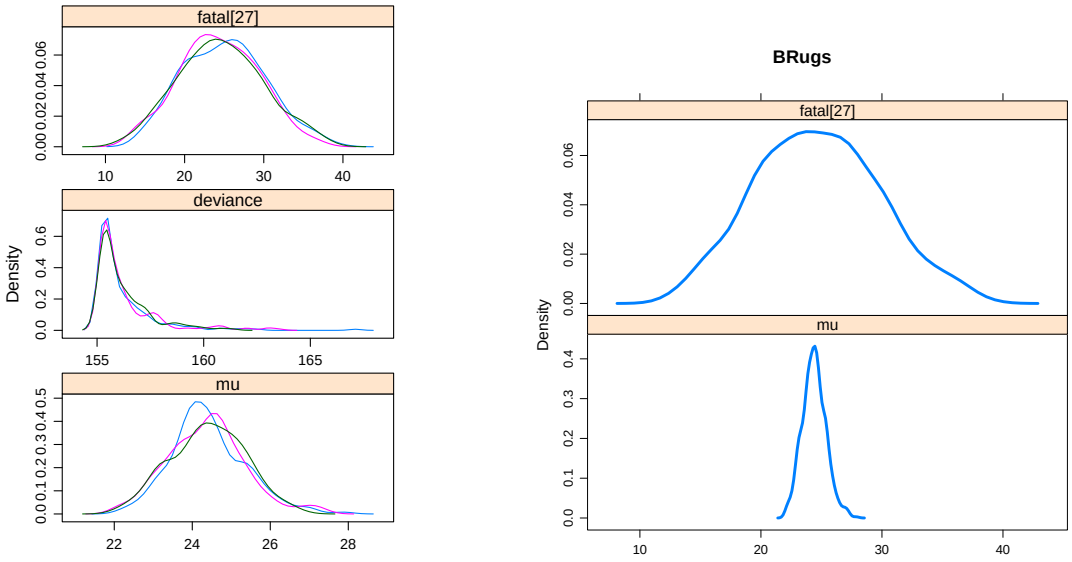

Figure 1.3: *Posterior densities; left panel is the default plot from* `densityplot`*, the right panel is the result from assembling the posterior from the three chains and doing a bit of grooming of the plot.*

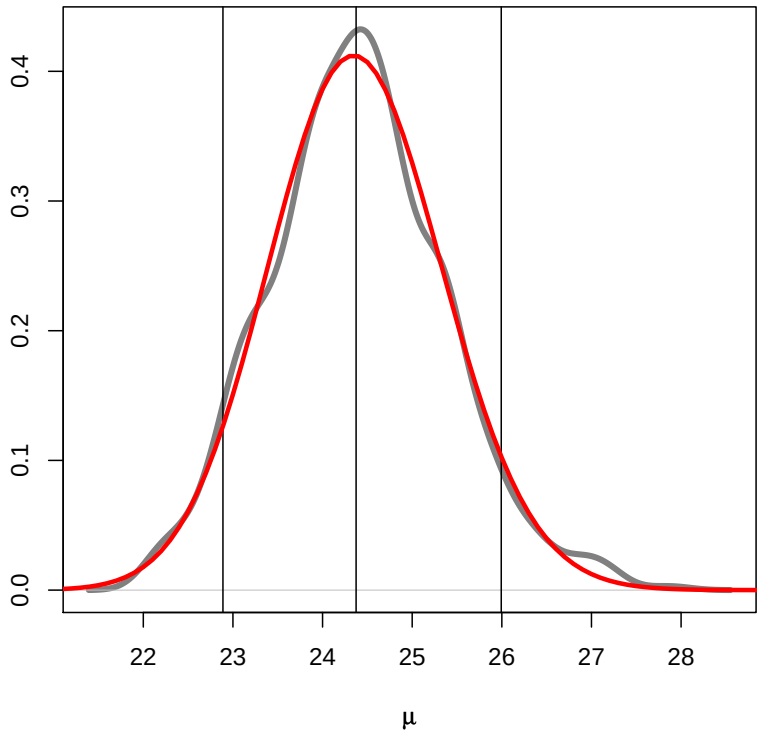

Figure 1.4: *Posterior density of the mean parameter with indication of 90% credibility interval.*

Chapter 2

Exercises

2.1 Bayesian inference in the binomial distribution

This exercise illustrates the prior to posterior calculations in the simple example of to inference about an unknown binomial probability, θ.

1. First, suppose that only a finite number of possible values for the true proportion θ are possible, *e.g.* $(\theta_1, \theta_2, \ldots, \theta_J)$, with prior probabilities $p(\theta_j)$, where $\sum_j p(\theta_j) = 1$. For a single Bernoulli trial $y \in (0, 1)$, the likelihood for each value for θ is given by

$$p(y|\theta_j) = {\theta_j}^y (1 - \theta_j)^{1-y},$$

   For an outcome y, Bayes' theorem combines the discrete prior distribution with the likelihood to generate posterior probabilities for the θ_j:

$$p(\theta_j|y) \propto {\theta_j}^y (1 - \theta_j)^{1-y} \times p(\theta_j),$$

   To get the proper posterior distribution, you have to normalize the r.h.s., that is divide by the sum.

   If have a binomial observation, i.e. x events out of n trials, then the posterior will be:

$$p(\theta_j|x) \propto \theta_j^x (1 - \theta_j)^{n-x} \times p(\theta_j).$$

   (a) Suppose a drug has an unknown true response rate θ, and for simplicity assume that θ can only take one of the values $\theta_1 = 0.2$, $\theta_2 = 0.4$, $\theta_3 = 0.6$ or $\theta_4 = 0.8$, and that we adopt the "neutral" position of assuming each value θ_j is equally likely, i.e. $p(\theta_j) = 0.25$ for each $j = 1, 2, 3, 4$.

   If we observe onle one person with a positive response ($y = 1$). How should our belief in the possible values be revised? Use this table to update from the prior to the posterior:

j	θ_j	Prior $p(\theta_j)$	Likelihood $p(y\|\theta_j)$	Likelihood × prior $p(y\|\theta_j)p(\theta_j)$	Posterior $p(\theta j\|y)$
1	0.2	0.25			
2	0.4	0.25			
3	0.6	0.25			
4	0.8	0.25			
	$\sum_j$	1.0			1.0

(b) If we instead of one patient had observations on $n = 20$ persons out which $x = 15$ had a positive response, how would the posterior look? Use that same table to complete the computations:

j	θ_j	Prior $p(\theta_j)$	Likelihood $p(y\|\theta_j)$	Likelihood × prior $p(y\|\theta_j)p(\theta_j)$	Posterior $p(\theta j\|y)$
1	0.2	0.25			
2	0.4	0.25			
3	0.6	0.25			
4	0.8	0.25			
	$\sum_j$	1.0			1.0

(c) Suppose we had given non-zero prior probability to the extreme values of $\theta = 0, 1$ (that is, the drug either never or always workes). The prior distribution is then on the six values $\theta_1 = 0$, $\theta_2 = 0.2$, $\theta_3 = 0.4$, $\theta_4 = 0.6$, $\theta_5 = 0.8$ or $\theta_6 = 1.0$, with $p(\theta_j) = 1/6$.
Describe *qualitatively* how the results in the table in part (a) would change if we used this discrete prior distribution on 6 values for θ for the same data, that is, 15 successes out of 20 trials. Uste this table for the calculations:

j	θ_j	Prior $p(\theta_j)$	Likelihood $p(y\|\theta_j)$	likelihood × prior $p(y\|\theta_j)p(\theta_j)$	Posterior $p(\theta j\|y)$
0	0.0	1/6			
1	0.2	1/6			
2	0.4	1/6			
3	0.6	1/6			
4	0.8	1/6			
5	1.0	1/6			
	$\sum_j$	1.0			1.0

(d) How would the results change if we used the data in the example in the module notes, that is, we had just one success from one trial?
You can use this table for the calculations:

j	θ_j	Prior $p(\theta_j)$	Likelihood $p(y\|\theta_j)$	likelihood × prior $p(y\|\theta_j)p(\theta_j)$	Posterior $p(\theta j\|y)$
0	0.0	1/6			
1	0.2	1/6			
2	0.4	1/6			
3	0.6	1/6			
4	0.8	1/6			
5	1.0	1/6			
	$\sum_j$	1.0			1.0

(*Hint*: It is not necessary to actually calculate the posterior probabilities explicitly. Try considering the value of the likelihood for each value of θ and the impact that the two new values of the likelihood for $\theta = 0$ and $\theta = 1$ will have on the calculations.

2. In the analysis above, for simplicity, we assumed that θ can could only take one of the values (0), 0.2, 0.4, 0.6, 0.8, (1).

Now suppose that previous experience with similar compounds has suggested that response rates between 0.2 and 0.6 could be feasible, with an expectation around 0.4. If we want a continuous prior distribution on the interval $(0, 1)$, we should choose one with mean 0.4 and say 95% of the probability mass in the interval (0.2,0.6), or more *ad hoc*, with a standard deviation of 0.1.

(a) We choose a Beta(a, b) as prior. From the properties of the beta distribution we know that mean m and standard deviation s are:

$$m = \frac{a}{a+b} \tag{2.1}$$

$$s = \sqrt{\frac{m(1-m)}{a+b+1}} \tag{2.2}$$

The expression in equation (2.2) can be rearranged to give $a + b = \big(m(1-m)/s^2\big) - 1$. Now use the target values $m = 0.4$ and $s = 0.1$ to obtain a value for $a + b$, and the formula for m to get separate values for a and b.

(b) Make a graph of the prior distribution for p, the success probability. The Beta-density is available in R as the function `dbeta`. You would need to type `?dbeta` to get the help function up.
(*Hint:* You can generate a vector of say 200 equidistantly spaced points between 0 and 1 by `seq(from=0,to=1,length=200)`.

(c) Suppose we observe $x = 15$ successes out of $n = 20$ trials. Make a graph of the likelihood for this observation. The binomial density is available in R as `dbinom`.

(d) From the prior distribution for the parameter and the likelihood we can form the posterior by taking the product. We know from lectures that the parameters of the beta distribution are updated to $[a^\star, b^\star]$ where $a^\star = a + x$ and $b^\star = b + (n - x)$.
Now make a third graph of the posterior for the success probability.

(e) Plot the three curves in one graph, using `par(mfrow=c(3,1))` before running the three plot statements.

(f) (*Complicated, but illustrative*) Pack the generation of the three graphs into an R-function that takes m, s (mean and standard deviation of the prior), x and n (the observed data) as arguments, and observe how the posterior changes when changing the prior and the data.

3. The French mathematician Pierre-Simon Laplace (1749–1827) was the first person to show definitively that the proportion of female births in the French population was less then 0.5, in the late 18th century, using a Bayesian analysis based on a uniform prior distribution (see Gelman *et al*; p.34). Suppose you were doing a similar analysis but you had more definite prior beliefs about the ratio of male to female births. In particular, if θ represents the proportion of female births in a given population, you are willing to place a Beta(100,100) prior distribution on θ.

(a) Show that this means you are more than 95% sure that θ is between 0.4 and 0.6, although you are ambivalent as to whether it is greater or less than 0.5.

(b) Now you observe that out of a random sample of 1,000 births, 511 are boys. What is your posterior probability that $\theta > 0.5$?

2.2 Simple linear regression with BUGS

The pupose of this exercise is to inrtoduce the use of BUGS as a machinery for estimation in standard statistical models. This is done using a simple linear regression example. The model we will be using is:

$$y_i = \alpha + \beta x_i + e_i, \qquad e_i \sim \mathcal{N}(0, \sigma^2)$$

assuming that the e_is are independent.

1. To make thinge easier, we use bogus data for the analysis:

   ```
   > x <- c(1,2,3,4,5,6)
   > y <- c(1,3,3,3,5,7)
   ```

   Plot them and make s standard linear regression using `lm()` from R: What are the estimates of intercept, slope and residual standard deviation in this model?

   Provide confidence intervals for α and β.

2. The next step is to use BUGS to estimate in the model. So referring to the section introducing BUGS, you should set up the following structures in R before invoking BUGS through the `bugs()` function:

   - Data — a list.
   - Initial values — a list of lists.
   - Parameters to monitor — a character vector.
   - A file with the BUGS program.

   In the program you must specify the model in terms of the three parameters of the model and the 6 observations of y and x. You should also specify the prior distributions of the parameters α, β of σ. Use uninformative priors for all three; that is normal priors with large variance for α and β, whereas a unform prior on some suitably large interval ([0,100], say) for σ is recommendable.

   Run the program for 20000 iterations with 10000 as burn-in.

3. Convert the result into a `mcmc.list` object using `as.mcmc.list(obj$sims.array)` and inspect the posterior using `summary`. Remember to load the `coda` package first. Compare the posterior medians and central 95% posterior intervals with the estimates and confidence intervals derived.

   How well do they agree?

4. Now try to do the same on a real dataset. In the `Epi` package is a datset, `births` which has data on 500 births in London, notably the birthweigst (`bweight`) and gestational age (`gestwks`). We will set up a rather naïve regression model with a linear relationship between x, number of gestational weeks and y birthweight.

   Now load the data and get the subset where the explanatory variable is non-missing:

   ```
   > library( Epi )
   > data( births )
   > births <- subset( births, !is.na(gestwks) )
   ```

   Re-use the set-up from the previous question to get classical regression estimates and estimates from the Bayesian machinery and compare them. Remember also to consider how the classically derived confidence intervals agree with the posterior central intervals.

2.3 Examples of the Gibbs sampler and Metropolis Hastings algorithm

1. Consider a single observation (y_1, y_2) from a bivariate normally distributed population with mean $\theta = (\theta_1, \theta_2)$ and known covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. With a uniform prior distribution on θ, the posterior distribution is

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} | y \sim \mathrm{N}\left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right).$$

Although it is simple to draw directly from the joint posterior distribution of (θ_1, θ_2), we set up the Gibbs sampler explicitly here for the purpose of illustration. To apply the Gibbs sampler to (θ_1, θ_2), we need the conditional posterior distributions.

(a) Use the properties of the multivariate normal distribution (either (A.1) or (A.2) on page 579 of **BDA**) to show that the relevant conditional distributions are

$$\begin{aligned} \theta_1 | \theta_2, y &\sim \mathrm{N}(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2), \\ \theta_2 | \theta_1, y &\sim \mathrm{N}(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2). \end{aligned}$$

(b) The Gibbs sampler proceeds by alternately sampling from these two normal distributions. In general we would say that the natural way to start the iterations would be with random draws from a normal approximation to the posterior distribution; of course, such draws would eliminate the need for iterative simulation in this trivial example!

Use the conditional distributions for θ_1 and θ_2 with $(y_1, y_2) = (0, 0)$ and $\rho = 0.8$ to set up a simple Gibbs sampler in R. Use two vectors, one for θ_1 called `theta1` and one for θ_2 called `theta2`, and start by setting the all the elements of each of `theta1` and `theta2` to 0:

```
> numsims <- 1000
> rho <- 0.8
> theta1 <- numeric(numsims)
> theta2 <- numeric(numsims)
```

Now amend the first value of `theta1` to -3 and sample a single value from the conditional distribution of θ_2 given θ_1 and set this as the first element of `theta2`:

```
> theta2[1] <- rnorm( 1, mean=rho*theta1[1], sd=sqrt(1 - (rho^2)) )
```

Now use a loop to iterate the process of sampling from the conditional distribution of θ_2 given θ_1 and *vice versa*:

```
> for(i in 2:numsims)
+ {
+ theta1[i] <- rnorm(1,mean=rho*theta2[i-1],sd=sqrt(1 -
+ (rho^2)))
+ theta2[i] <- rnorm(1,mean=rho*theta1[i],sd=sqrt(1 -
+ (rho^2)))
+ }
```

Generate 1000 values for each of θ_1 and θ_2 using the Gibbs sampling routine from part (b) of the question. Calculate the sample mean and standard deviation of the final 500 realised values for each of θ_1 and θ_2. Show that these empirical values for the mean and standard deviation are close to the theoretical values for the posterior marginal distributions of θ_1 and θ_2 based on the joint posterior distribution displayed above:

```
> mean(theta1[501:1000])
> mean(theta2[501:1000])
> sqrt(var(theta1[501:1000]))
> sqrt(var(theta2[501:1000]))
```

Also check that the correlation between the two sequences is close to the true value of 0.8:

```
> cor(theta1[501:1000], theta2[501:1000])
```

2. We can also use the Metropolis-Hasting algorithm to sample from the posterior distribution. For the proposal distribution $h()$ we use the uncorrelated bivariate normal distribution. Implement this in R by working through the following.

   Set the correlation to $\rho = 0.7$, the number of simulation `nsim` to 1000, initialise a matrix `ans` with 1000 rows and 2 columns that will hold the results of the simulation and set up the 2×2 correlation matrix `Sigma` and its inverse `SigmaInv`:

```
> rho  <- 0.7
> nsim <- 1000
> ans  <- matrix(NA, nr=nsim, nc=2)
> Sigma     <- matrix(c(1,rho,rho,1), nr=2)
> SigmaInv  <- solve(Sigma)
```

   We start the simulation at `x1` = `x2` = 30 and set up a vector `xcurr` that holds the current values of `x1` and `x2`:

```
> x1 <- x2 <- 30
> xcurr <- c(x1,x2)
```

   Initialise an "acceptance vector" called `accept` to 0 and the standard deviation `sigma` of the proposal distribution to 2. Run `nsim` iterations and at each iteration, generate a proposal called `xprop` by adding a normal random variate with mean 0 and standard deviation 2 to the current value. Calculate the log-likelihood for both the current and proposed values and accept this with the appropriate probability. If the proposal is accepted, the correspondign component of the `accept` vector is set to 1 (in fact "TRUE"), otherwise 0 ("FALSE"):

```
> accept <- numeric(nsim)
> sigma <- 2
> for (ii in 1:nsim){
+   xprop  <- xcurr + rnorm(2, mean = 0, sd = sigma)
+
+   logkxprop <- - t(xprop) %*% SigmaInv %*% xprop /2
+   logkxcurr <- - t(xcurr) %*% SigmaInv %*% xcurr /2
+
+   alpha <- min(1, exp(logkxprop-logkxcurr))
+   u <- rnorm(1)
+
+   if ( accept[ii] <- (u<alpha) ){
+     xaccept <- xprop
+   } else {
+     xaccept <- xcurr
+   }
+
+   ans[ii,] <- xaccept
+   xcurr <- xaccept
+ }
> cat("Accepted proposals: ", sum(accept)/nsim, "\n")
```

   Now plot all samples:

```
> pairs(ans)
```

Plot the two series of values (`x1` and `x2`) to determine the number of iterations that we need to use as the burn-in:

```
> matplot(ans, type='l')
```

It looks like it is sufficient to discard the first 100 samples as the burn in:

```
> pairs(ans[-(1:100),])
```

We can check dependencies among each of the series for `x1` and `x2` using the autocorrelation functions `pacf` (for *partial* autocorrelation) and `acf`:

```
> par( mfrow=c(2,2) )
> pacf(ans[,1])
> pacf(ans[,2])
>  acf(ans[,1])
>  acf(ans[,2])
```

You should investigate the effect of changing

(a) The value of the correlation parameters ρ.

(b) The mean of the proposal distribution.

(c) The standard deviation of the proposal distribution.

3. It's instructive to compare the bivariate sampler above to a single component Metropolis–Hastings sampler where the proposal for $h(x_2|x_1^t, x_2^t)$ is $x_2 = x_2^t + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$ for some choice of σ^2 and likewise for x_1. The set up is the same:

```
> rho  <- 0.7
> nsim <- 1000
> ans  <- matrix(NA, nr=nsim, nc=2)
> x1 <- x2 <- 30
> xcurr <- c(x1,x2)
```

We now need two counters, one for each component of the vector containing the values of `x1` and `x2`. We need to calculate the log-likelihood of the conditional distribution of `x1` given `x2` for both the current and proposed value of `x1` and proposal (the quantities `logpx1prop` and `logpx1`, along with the unconditional log-likelihoods `hx1prop` and `hx1`, all of which are used in generating the ratio governing the acceptance probability. We run through the same routine for `x2`.

```
> accept1 <- accept2 <- numeric(nsim)
> sigma <- 5
> for (ii in 1:nsim){
+
+   # Update x1:
+   x1prop <- rnorm(1, mean=x1, sd=sigma)
+
+   logpx1prop  <- -(x1prop-rho*x2)^2/(1-rho^2)
+   logpx1      <- -(x1-rho*x2)^2/(1-rho^2)
+
+   hx1prop <- dnorm(x1prop, mean=x1, sd=sigma)
+   hx1     <- dnorm(x1, mean=x1prop, sd=sigma)
+
+   alpha <- min(1, exp(logpx1prop-logpx1)*(hx1/hx1prop))
+   u     <- rnorm(1)
```

```
+
+   if ( accept1[ii] <- (u<alpha) ){
+     x1 <- x1prop
+   }
+
+   # Update x2:
+   x2prop <- rnorm(1, mean=x2, sd=sigma)
+
+   logpx2prop  <- -(x2prop-rho*x1)^2/(1-rho^2)
+   logpx2      <- -(x2-rho*x1)^2/(1-rho^2)
+
+   hx2prop <- dnorm(x2prop, mean=x2, sd=sigma)
+   hx2     <- dnorm(x2, mean=x2prop, sd=sigma)
+
+   alpha <- min(1, exp(logpx2prop-logpx2)*(hx2/hx2prop))
+   u     <- rnorm(1)
+
+   if ( accept2[ii] <- (u<alpha) ){
+     x2 <- x2prop
+   }
+   ans[ii,] <- c(x1,x2)
+ }
> cat("Accepted proposals, x1: ", sum(accept1)/nsim, "x2:", sum(accept2)/nsim, "\n")
```

Once again we can plot all the samples:

```
> pairs(ans)
```

Check the number of iterations that we need to discard as a burn-in:

```
> matplot(ans, type='l')
```

Let's discard the first 100 samples:

```
> pairs(ans[-(1:100),])
```

Have a look at the cumulative acceptance probabilities for `x1` and `x2`:

```
> plot( 1:nsim,cumsum(accept1)/1:nsim, ylim = c(0,1), pch = "",
+       xlab = "Iteration Number", ylab = "Probability")
> lines(1:nsim,cumsum(accept1)/1:nsim, ylim = c(0,1), lwd = 3)
> title(main = "Cumulative acceptance probability", cex = 0.5)

> plot( 1:nsim,cumsum(accept2)/1:nsim, ylim = c(0,1), pch = "",
+       xlab = "Iteration Number", ylab = "Probability")
> lines(1:nsim,cumsum(accept2)/1:nsim, ylim = c(0,1), lwd = 3)
> title(main = "Cumulative acceptance probability", cex = 0.5)
```

Also let's plot the two series `x1` and `x2` against each other (change the value of the standard deviation in the simulations above to see the jumps get bigger or smaller):

```
> plot(ans[,1],ans[,2],ylim = c(-50,50),xlim = c(-50,50), xlab = "x1", ylab = "x2")
> lines(ans[,1],ans[,2],lwd = 1)
> title(main = "Metropolis-Hastings sampler s.d. = 2")
```

Finally check the dependencies within each of the `x1` and `x2` series:

```
> par( mfrow=c(2,2) )
> pacf(ans[,1])
> pacf(ans[,2])
> acf(ans[,1])
> acf(ans[,2])
```

Consider the following questions:

(a) What the is cumulative acceptance probability after 1000 simulations? How many simulations are before the acceptance ratio stabilises?

(b) Explore how changing the standard deviation of the proposal distributions alters

   i. the cumulative acceptance ratio,

   ii. the number of iterations required to achieve convergence and a stable acceptance ratio,

   iii. the visual appearance of the sample path of the bivariate plot.

2.4 Estimating a rate from Poisson data

Asthma deaths in Australia (*cf* Section 2.7 of *Bayesian Data Analysis* pages 53-55).

The death toll for asthma in Australia in 2002 was 397, down from 422 in 2001 and 454 in 2000 (source: National Asthma Council of Australia, `www.nationalasthma.org.au`). This latest figure represents a rate of very close to 400 in 20 million, or 2 cases per 100,000 persons per year, corresponding to $\theta = 2$ in the example. The observed value in the example was 3 asthma deaths in a population of 200,000, an observed rate of 1.5 deaths per 100,000 persons per year.

1. What's the posterior probability, using the gamma(3.0,5.0) prior in the example, that the true rate in the hypothetical city of 200,000 people is actually higher than the observed Australian rate of about 2 deaths per 100,000 persons per year? The relevant `BUGS` code can be found in the file `asthma.odc`.

2. Use the `BUGS` code in `asthma.odc` from question 1 as the basis for preparing a second set of `BUGS` code to incorporate the Australia figures for 2002, that is, 397 deaths from 20 million people, in addition to the existing figures of 3 deaths in the hypothetical population of 200,000. You will need to recast the nodes `y`, `lambda`, `theta` and `n` as arrays of dimension 2 (so `y` would actually be `y[1]` and `y[2]`, where the first element of each array refers to the original hypothetical data and the second element refers to the Australian data). Set up a separate additional node to monitor when the difference in the sampled values of θ_1 and θ_2 is bigger than zero. Compile the `BUGS` model and use it to calculate the posterior probability that the difference $\theta_1 - \theta_2 > 0$, where θ_1 corresponds to the original rate parameter in part (a) and θ_2 corresponds to the Australian rate.

   You can use the `BUGS` code in the files from previous practicals to get some ideas as to how to set up the relevant arrays, `for` loop and posterior probability nodes based on the `step` function.

3. Why wouldn't you expect the answer to be much different from the answer we got in part (a) where we assumed the Australian rate to be exactly 2 deaths per 100,000?

4. Suppose that we had the following additional data on the number of asthma deaths in Australia (source: National Asthma Council of Australia, `www.nationalasthma.org.au`), availbel in the course material as `asthma.dat`:

Year	Asthma Deaths
1997	499
1998	481
1999	424
2000	454
2001	422
2002	397
2003	314
2004	311
2005	318
2006	402

   Extend the model in the previous questions to accommodate these new data. The number of deaths in 2006 (402) is much higher than the previous three years where the number of

deaths was about 300 - how could we check this formally? For some ideas have a look at the airline example in exercises 2.13 and 3.12 of Bayesian Data Analysis and exercise 6 from this course.

2.5 Estimating the speed of light

Simon Newcomb set up an experiment in 1882 to measure the speed of light. Newcomb measured the amount of time required for light to travel 7442 metres. The measurements are availble in the file `newcomb.r`, which should be `source`d to get the data into R:

```
> source("../data/newcomb.r")
> newcomb

 [1]  28  26  33  24  34 -44  27  16  40  -2  29  22  24  21  25  30  23  29  31
[20]  19  24  20  36  32  36  28  25  21  28  29  37  25  28  26  30  32  36  26
[39]  30  22  36  23  27  27  28  27  31  27  26  33  26  32  32  24  39  28  24
[58]  25  32  25  29  27  28  29  16  23
```

A histogram of Newcomb's 66 measured is shown in figure 2.1. There are two unusually low measurements and then a cluster of measurements that are approximately symmetrically distributed. We (inappropriately!) apply the normal model, assuming that all 66 measurements are independent draws from a normal distribution with mean μ and variance σ^2. The main substantive goal is posterior inference for μ. The outlying measurements do not fit the normal model, an issue that we pursue briefly in question 4. The sample mean of the $n = 66$ measurements is $\overline{y} = 26.2$, and the sample standard deviation is $s = 10.8$.

1. Assuming the non-informative prior distribution $p(\mu, \sigma^2) \propto (\sigma^2)^{-1}$ (which is equivalent to a joint uniform prior distribution on $(\mu, \log \sigma)$), the posterior distribution of μ has the form

$$\left| \frac{\mu - \overline{y}}{s/\sqrt{n}} \right| \sim t_{n-1}. \tag{2.3}$$

   Note that only μ is unknown in the expression above since we are conditioning on the observed values of the sample mean $\overline{y}$, the sample standard deviation s and the sample size n. Use this distributional result to calculate a 95% central posterior interval for μ.

2. The posterior interval can also be obtained by simulation. Following the factorisation of the posterior distribution given in lectures as

$$\begin{aligned} p(\mu|\sigma^2, y) &\sim \mathrm{N}(\overline{y}, \sigma^2/n) \\ p(\sigma^2|y) &\propto (\sigma^2)^{-(n+1)/2} \exp\left(-\frac{(n-1)s^2}{2\sigma^2}\right), \end{aligned}$$

   which is a scaled inverse-χ^2 density:

$$p(\sigma^2|y) \sim \chi^{-2}(n-1, s^2),$$

   we first draw a random value of $\sigma^2 \sim \chi^{-2}(65, s^2)$ as $65s^2$ divided by a random draw from the χ^2_{65} distribution. Then given this value of σ^2, we draw μ from its conditional posterior distribution, $\mathrm{N}(26.2, \sigma^2/66)$.

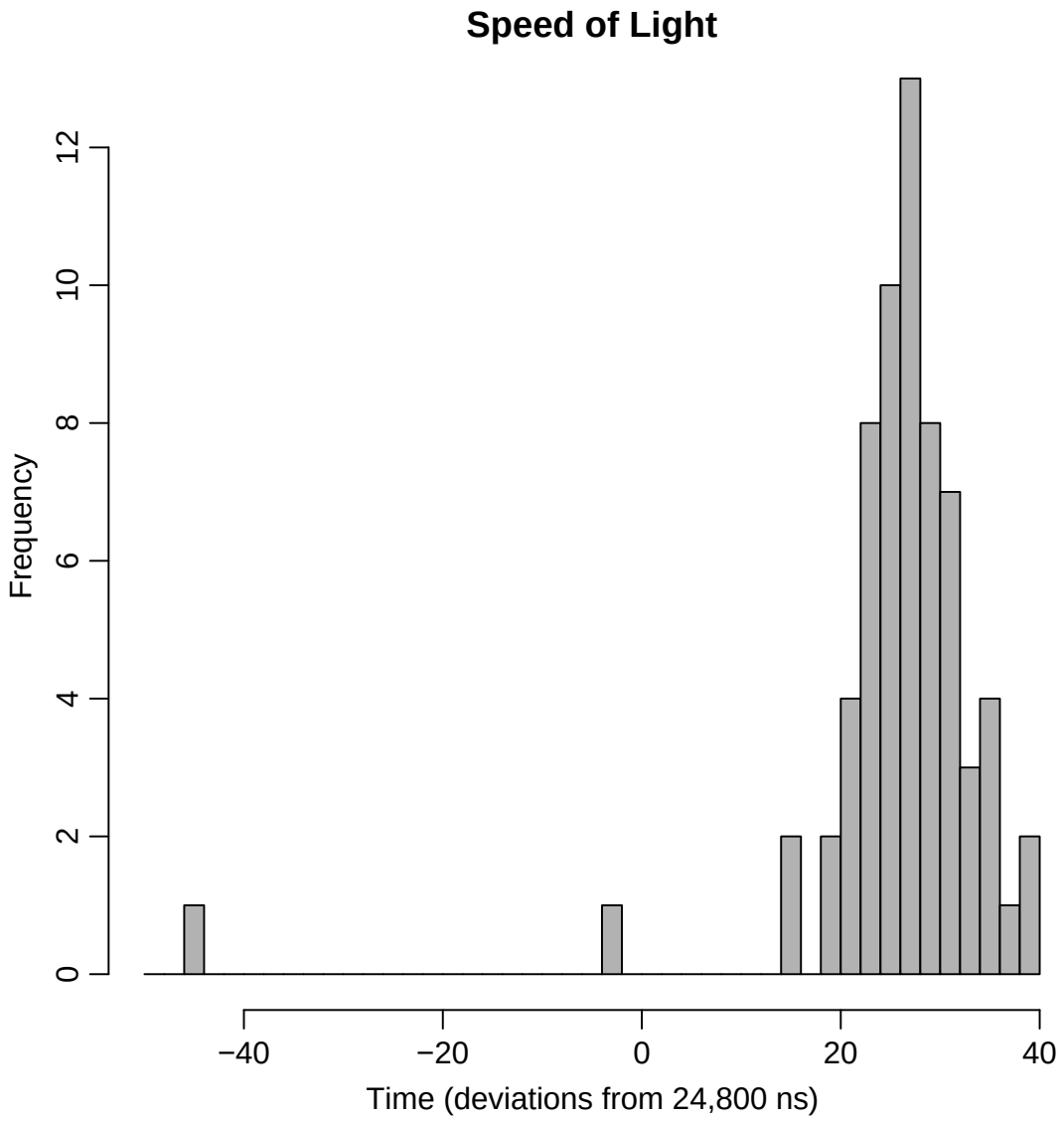

Figure 2.1: Histogram of Simon Newcomb's measurements for estimating the speed of light, from Stigler SM. (1977). Do robust estimators work with real data? (with discussion). *Annals of Statistics* **5**, 1055-1098. The data are times for light to travel a fixed distance, recorded as deviations from 24,800 nanoseconds.

Use the following computing code (contained in the file `speed_of_light.R`) to carry out these simulation steps (for 1,000 iterations) in `R` and generate a vector of sampled values `mu`. You can type `summary(mu)` to get a summary of the values in the vector `mu` (including the mean), and type `sort(mu)` to display the ordered values in the vector `mu`; the 25^{th} and 975^{th} values provide estimates of the limits of the 95% posterior credible interval for μ (these can be accessed directly by typing `sort(mu)[25]` and `sort(mu)[975]`). The `R` code:

```
> ybar <- mean(newcomb)
> s <- sqrt(var(newcomb))
> n <- 66
> numsims <- 1000
> mu <- numeric(length = numsims)
> sigma2 <- numeric(numsims)
> for( i in 1:numsims )
+    {
+    sigma2[i] <- (65*(s^2))/(rchisq(1,n-1,ncp=0))
+    mu[i] <- rnorm(1, mean = ybar, sd = sqrt(sigma2[i]/n))
+    }
```

3. Check the results in questions 1 and 2 using the `BUGS` code in the file `speed_of_light.odc` which represents the same model and can be used to simulate from the posterior distributions for μ and σ.

4. Based on the currently accepted value of the speed of light, the "true value" for μ in Newcomb's experiment is 33.0, which not only falls outside our 95% interval from questions 1 and 2 but has a "z-score" based on the posterior distribution for μ of about 5; so values as large as this or larger attract very little posterior probability under our model for the data. This reinforces the fact that posterior inferences are only as good as the model and the experiment that produced the data.

   One way we can check the suitability of the model is to amend the `BUGS` code from question 3 so that it *generates* a vector `y.pred` of 66 observations from the normal distribution with the current sampled values of μ and σ. We can then ask `BUGS` to retain the smallest value from the vector `y.pred`, generating a distribution of minimum measurements for a sample of size $n = 66$.

   Open the file `BUGS` file `speed_of_light_pred.odc` and identify the changes that have been made to the original `BUGS` file from question 3 (or, even better, try to make these changes yourself before looking at the file!). Compile the model and run for 10,000 burn-in iterations and 10,000 further iterations, summarise the distribution of the "smallest" value and comment on the likelihood of observing the two negative observed values (-2 and -44).

   See chapter 6 in Gelman *et al.* for an extensive discussion of such "posterior predictive checking", in particular a more detailed treatment of the problem discussed here in section 6.3 pages 160-161.

2.6 Modelling the rate of airline fatalities 1976 to 2001

This exercise is based on exercises 2.13 and 3.12 from Gelman *et al.*. The original exercise has been extended to include additional data from 1986 to 2001. It is useful to read the partial solution to the original exercise 2.13 that appears in the most recent solutions file on Andrew Gelman's website, which is available as a PDF.

The data is available in the text file `airline.txt` with column names in the first line, aimed a reading into `R`. It is easier to work with distances in units of 10^{11} miles, which is how the passenger miles and accident rate data are presented in both source files (`.odc` and `.txt`).

The file `sol6a.R` contains an R-program that read data, produces all the relevant plots suggested in the following exercise. The R-file also contains specifications of the models used in `BUGS` and calls to `WinBUGS` using the package `R2WinBUGS`.

Table 2.1: *Worldwide airline fatalities, 1976–2001. "Passenger miles" are in units of* 10^{11} *and the "Accident rate" is the number of fatal accidents per* 10^{11} *passenger miles. Source: International Civil Aviation Organization, Montreal, Canada (*`www.icao.int`*)*

Year	Fatal accidents	Passenger miles	Accident rate
1976	24	3.863	6.213
1977	25	4.300	5.814
1978	31	5.027	6.167
1979	31	5.481	5.656
1980	22	5.814	3.784
1981	21	6.033	3.481
1982	26	5.877	4.424
1983	20	6.223	3.214
1984	16	7.433	2.152
1985	22	7.107	3.096
1986	22	9.100	2.418
1987	25	10.000	2.500
1988	29	10.600	2.736
1989	29	10.988	2.639
1990	27	10.880	2.482
1991	29	10.633	2.727
1992	28	11.956	2.342
1993	33	12.343	2.674
1994	27	13.011	2.075
1995	25	14.220	1.758
1996	24	16.371	1.466
1997	26	15.483	1.679
1998	20	18.080	1.106
1999	21	16.633	1.263
2000	18	18.875	0.954
2001	13	19.233	0.676

1. The simplest model: All years look the same.

   (a) Assume that the numbers of fatal accidents in each year are independent with a

Poisson(θ) distribution. Set a (noninformative) gamma prior distribution for θ and determine theoretically using the results in lectures the posterior distribution based on the data from 1976 through 2001.

(b) In this case it is also possible to determine theoretically the predictive distribution for the number of fatal accidents in 2002 - what is it? (See Section 2.7 page 53 of Gelman *et al.*).

(c) How can we use the posterior distribution for θ and the assumption about the distribution of the number of fatal accidents to construct a two-stage process to draw samples from the predictive distribution for the number of fatal accidents in 2002?

(d) If we set up a node in `BUGS` for year 2002 (*i.e.* adding an extra component to the data array for years 1976 to 2001 as has been done in the computing code provided) with the number of fatal accidents declared as "`NA`" (missing) will cause `BUGS` to draw from the predictive distribution for this node. What is the 95% predictive interval for the number of fatal accidents in 2002?

2. A model with constant *rate* of fatal airline crashes.

(a) Now assume that the numbers of fatal accidents in each year follow independent Poisson distributions with a mean proportional to the number of passenger miles flown. Using the same noninformative prior distribution for θ determine the posterior distribution of the rate, i.e. accidents per passenger miles.

(b) Modify your `BUGS` code from the previous question to accomodate this model, and use it to generate a 95% predictive interval for the number of fatal accidents in 2002 under the assumption that 2×10^{12} passenger miles were flown that year.
(*Hint:* Note that you cannot stick an expression in as an argument to a distribution in `BUGS`; an expression as `fatal[i] dpois(lambda*miles[i])` will cause an error, so you will have to construct nodes for the mean, *e.g.* `mu[i] <- lambda * miles[i]; fatal[i] dpois( mu[i] ).`)

3. We now expand the model by assuming that the number of fatal accidents in year t follows a Poisson distribution with mean $\alpha + \beta t$, *i.e.* independent of passengar miles but merely linearly decreasing by time.

(a) Plot the number of fatal accidents each year over time to see that this was a dubious assumption even with the original data and is certainly not reasonable in light of the new data - why?

(b) Moreover, a linear function of time t has the potential to generate negative values unless the parameters α and β are constrained - why is this a problem?

4. It would be more satisfactory to assume that the number of fatal accidents $y(t)$ in year t where $m(t)$ passenger miles were flown follows a Poisson distribution with mean $\big(\exp(\alpha + \beta t)\big)m(t)$. This is a generalised linear model with canonical (log) link:

$$\mathrm{E}\big(y(t)|t, m(t)\big) = \big(\exp(\alpha + \beta t)\big)m(t) \tag{2.4}$$

$$\log\Big(\mathrm{E}\big(y(t)|t, m(t)\big)\Big) = \alpha + \beta t + \log(m(t)) \tag{2.5}$$

(a) Calculate crude estimates and uncertainties for (α, β) using linear regression based on the relationship described above in equation (2.5), i.e. using the log-rates as reponse variable.

(b) Fit the generalized linear model using `glm` in R.

(c) Use the estimates from the maximum likelihood estimation as initial values to run the model in `BUGS` and to generate samples from the posterior distribution of α and β.

(d) Use the `xyplot.mcmc.list` function to check the mixing of the chains for α and β.

(e) Use the `densityplot.mcmc` function to display smoothed marginal posterior densities for α and β based on the sampled values of α and β. Also, make a scatter-plot showing the joint posterior distribution of α and β.

(f) Plot the posterior density for the *expected number* of fatal accidents in 2002, $\big(\exp(\alpha + 2002\beta)\big) \times m(2002)$ where we again assume the number of miles flown in 2002 is 2×10^{12}.

(g) Obtain the 95% predictive distribution interval for the *number* of fatal accidents in 2002.

(h) How would you define and derive the posterior predictive distribution of the number of fatalities in 2002, from the maximum likelihood approach?

2.7 Assessing convergence using the Gelman-Rubin diagnostic — Using coda in R

1. Compile the model in the file `schools.odc` for the SAT coaching data presented in *Bayesian Data Analysis* and also as an example in the documentation for the `bugs()` function in the `R2WinBUGS` package in `R`.

   The `schools` example data are a part of the `R2winBUGS` data:

```
> library( R2WinBUGS )
> data( schools )
> schools

  school estimate   sd
1      A    28.39 14.9
2      B     7.94 10.2
3      C    -2.75 16.3
4      D     6.82 11.0
5      E    -0.64  9.4
6      F     0.63 11.4
7      G    18.01 10.4
8      H    12.16 17.6
```

   Use `BUGS` in R, and ensure that you are running multiple chains. The standard output displays the Gelman-Rubin diagnostic (the potential scale reduction factor $\hat{R}$) for each monitored node, so after `BUGS` has completed the specified number of iterations you should see values of $\hat{R}$ for μ_θ (`mu.theta`), σ_θ (`sigma.theta`) and for each of the components of $\theta = (\theta_1, \theta_2, \ldots, \theta_8)$ (`theta[1]`, `theta[2]`, ... , `theta[8]`).

   (a) Run the compiled model 10 times for just 100 iterations, and make a note of the potential scale reduction factor $\hat{R}$ for each monitored node on each occasion. You can either do this manually by observation, estimating the values of $\hat{R}$ from the graphical output, or access the calculated values by displaying `schools.sim$summary`.

      (i) Which nodes have values of $\hat{R}$ close to 1 and thus appear to have reached convergence?

      (ii) Which nodes have values of $\hat{R}$ that are "not close to 1" (bigger than 1.2) and therefore require more iterations to reach convergence?

      (iii) Do all nodes have values of $\hat{R}$ that are either all close to 1 or all not close to 1, or are there some nodes that have "low" values of $\hat{R}$ for some runs and large values on others? This raises the question of whether we need to be concerned about *sampling variation* in $\hat{R}$.

   (b) Repeat part (a) and steps (i) to (iii) above using 300 iterations per chain.

   (c) Repeat part (a) and steps (i) to (iii) above using 500 iterations per chain.

   Comment on the improved convergence using an increasing number of iterations. Do you think 1,000 iterations, as used in the original practical session was sufficient to ensure convergence of all nodes?

2. Load and install the `coda` package in R. Use the output dataframe `schools.sim` (in particular the list `schools.sim$sims.array` containing the values for each monitored node in each simulated chain), which is created after compiling and running the `schools` model in `BUGS` through R, to create some `mcmc` and `mcmc.list` objects. Use these objects as inputs

to the diagnostic procedures in `coda`, such as `gelman.diag`, `gelman.plot`, `geweke.diag`, `geweke.plot`, `hiedel.diag` and `raftery.diag` to gain an overview of the convergence diagnostics provided by the `coda` package.

2.8 Meta-analysis of clinical trial data

This example, from Spiegelhalter *et al.* (2004), is described in more detail in Higgins and Spiegelhalter (2002). The numbers have been re-worked from the original example using the same raw data to provide consistency with the textbook for this subject, since Spiegelhalter *et al.* (2004) use a "continuity correction" (adding $frac12$ to the numerator and denominator) when calculating estimated odds ratios and the standard deviation of their logarithm which is not used by Gelman *et al.* in *Bayesian Data Analysis*.

Epidemiology, animal models and biochemical studies suggested intravenous magnesium sulphate may have a protective effect after acute myocardial infarction (AMI), particularly through preventing serious arrhythmias. A series of small randomised trials culminated in a meta-analysis (Teo *et al.* (1991)) which showed a highly significant ($P < 0.001$) 55% reduction in odds of death. The authors concluded that "further large scale trials to confirm (or refute) these findings are desirable", and the LIMIT-2 trials (Woods *et al.* (1992)) published results showing a 24% reduction in mortality in over 2000 patients. An editorial in *Circulation* subtitled "An effective, safe, simple and inexpensive treatment" (Yusuf *et al.* (1993)) recommended further trials to obtain "a more precise estimate of the mortality benefit". Early results of the massive ISIS-4 trial pointed, however, to a lack of any benefit, and final publication of this trial on over 58,000 patients showed a non-significant adverse mortality effect of magnesium. ISIS-4 found no effect in any subgroups and concluded that "overall, there does not now seem to be any good clinical trials evidence for the routine use of magnesium in suspected acute MI" (Collins *et al.* (1995)).

The aim of the re-analysis presented here is to investigate how a Bayesian perspective might have influenced the interpretation of the published evidence on magnesium sulphate in AMI available in 1993.

We present here a meta-analysis of randomised trials. The outcome measure is the odds ratio for in hospital mortality, with odds ratios less than 1 favouring magnesium. We outlined three approaches in the lecture for modelling results from multiple trials but we'll concentrate here on (a) a "pooled" analysis assuming identical underlying effects and (b) a random-effects analysis assuming exchangeable treatment effects, ignoring the third option (c) a fixed-effect analysis assuming independent, unrelated effects where estimates of a trial-specific effect for each trial are obtained using only data from that trial.

We begin with an empirical Bayes analysis, using estimates of the overall mean μ and the between-study standard deviation τ, in order to use the formula-driven normal posterior analysis described in the lectures. For both the pooled- and fixed-effects analysis we assume a uniform prior for the unknown effects on the log(OR) scale. The empirical Bayes analysis does not use any prior distributions on the parameters μ and τ (although the estimate for μ is equivalent to assuming a uniform prior on the log(OR) scale).

We also conduct a full Bayesian analysis by placing prior distributions on both the overall treatment effect μ and the between-treatments standard deviation τ. A sensitivity analysis is performed using a "neutral" prior for μ centred on "no effect", which allows for scepticism about large effects.

It is straightforward to conduct most of these analysis below using a spreadsheet, since the conjugate normal analysis allows us to work with closed-form expressions. Data for this set of exercises is in `mag.xls` or `mag.RData`, and `BUGS` codes can be found in `mag.odc`.

1. Calculate the pooled estimate $\hat{\mu}$ and a 95% confidence interval for assumed common treatment effect μ using the formula in the lecture notes.

2. The unknown hyperparameters μ and τ may be estimated directly from the data - this is known as the "empirical Bayes" approach as it avoids specification of the prior distributions

for μ and τ. There are a variety of techniques available as they form part of classical random-effects meta-analysis (Sutton *et al.* (2000); Whitehead (2002)). However, the simplest is the "method-of-moments" estimator (DerSimonian and Laird (1986))

$$\hat{\tau}^2 = \frac{Q-(J-1)}{\sum_{j=1}^{J} 1/\sigma_j^2 - \frac{\sum_{j=1}^{J} 1/\sigma_j^4}{\sum_{j=1}^{J} 1/\sigma_j^2}} \tag{2.6}$$

where $J = 8$ is the number of trials and Q is the test for homogeneity

$$Q = \sum_{j=1}^{J} \frac{(y_j - \hat{\mu})^2}{\sigma_j^2}. \tag{2.7}$$

If $Q < J-1$ then $\hat{\tau}^2$ is set to zero and complete homogeneity is assumed.

(a) Calculate Q and hence the P-value for the test of homogeneity based on the null distribution for Q which is chi-squared with $J-1$ degrees of freedom.

(b) Use the calculated value of Q and the formula in equation 2.6 to calculate the value of $\hat{\tau}^2$. Figure 2.2 shows the *profile* likelihood (see the lecture), which summarises the support for different values of τ. Note that the maximum likelihood estimator of τ^2 is zero although the profile likelihood suggests reasonable support for values of τ as large as 1.

3. Let's now get `BUGS` to perform the random-effects analysis of the same data (although you may continue to perform the calculations using the Microsoft Excel spreadsheet if you wish!). Fix the value of $\hat{\tau}^2$ at its method-of-moments estimate, and run the `BUGS` code with an approximately uniform prior distribution for μ. Output the posterior summary statistics for μ and $\theta = (\theta_1, \theta_2, \ldots, \theta_8)$.

   (a) Compare the posterior means for the components of θ to the empirical log odds ratios in table 2.2 of the question sheet.

   (b) What is the posterior mean and 95% credible interval of the "average" effect μ and how does it compare to the pooled effect? The results are shown in figure 2.3.

4. The random-effects analysis above is not a full Bayesian analysis since it uses no prior distribution for $\hat{\tau}^2$ (other than the trivial degenerate prior that places 100% of the probability mass at the data-driven method-of-moments estimate). Change the prior distribution for τ to uniform on (0,1000) and re-run the model. How does this affect the results?

5. The meta-analyses above, whether a pooled- or random-effects analysis, finds a "significant" benefit from magnesium. The apparent conflict between this finding and the results of the ISIS-4 mega-trial have led to lengthy dispute, briefly summarised in Higgins and Spiegelhalter (2002). We consider now the robustness of the meta-analysis results to the choice of prior distribution, by performing a "credibility analysis" that checks whether the findings are robust to a reasonable expression of prior "scepticism" concerning large benefits. Re-do the analysis using the same vague prior distribution for τ, but now with a prior distribution for μ that is normal with mean 0 (so it is centred on the null value) and standard deviation 0.40, so that there is about a 5% chance that the true odds ratio is less than 0.5, that is, we're sceptical about a large benefit of the treatment. How does changing to this new prior distribution alter the conclusions of the meta-analysis?

Trial	Magnesium group		Control group		Estimated log(OR) y_k	Estimated SD s_k	Shrinkage B_k
	deaths	patients	deaths	patients			
Morton	1	40	2	36	-0.83	1.25	0.90
Rasmussen	9	135	23	135	-1.06	0.41	0.50
Smith	2	200	7	200	-1.28	0.81	0.80
Abraham	1	48	1	46	-0.04	1.43	0.92
Feldstedt	10	150	8	148	0.22	0.49	0.59
Schechter	1	59	9	56	-2.41	1.07	0.87
Ceremuzynski	1	25	3	23	-1.28	1.19	0.89
LIMIT-2	90	1159	118	1157	-0.30	0.15	0.11

Table 2.2: Summary data for magnesium meta-analysis, showing estimated odds ratios, log(odds ratios) (y_j), standard deviations for log(odds ratios) (σ_j) and shrinkage coefficients $B_j = \sigma_j^2/(\sigma_j^2 + \hat{\tau}^2)$. $\hat{\tau}$ is taken to be the method-of moments estimate 0.41 from equation (2.6).

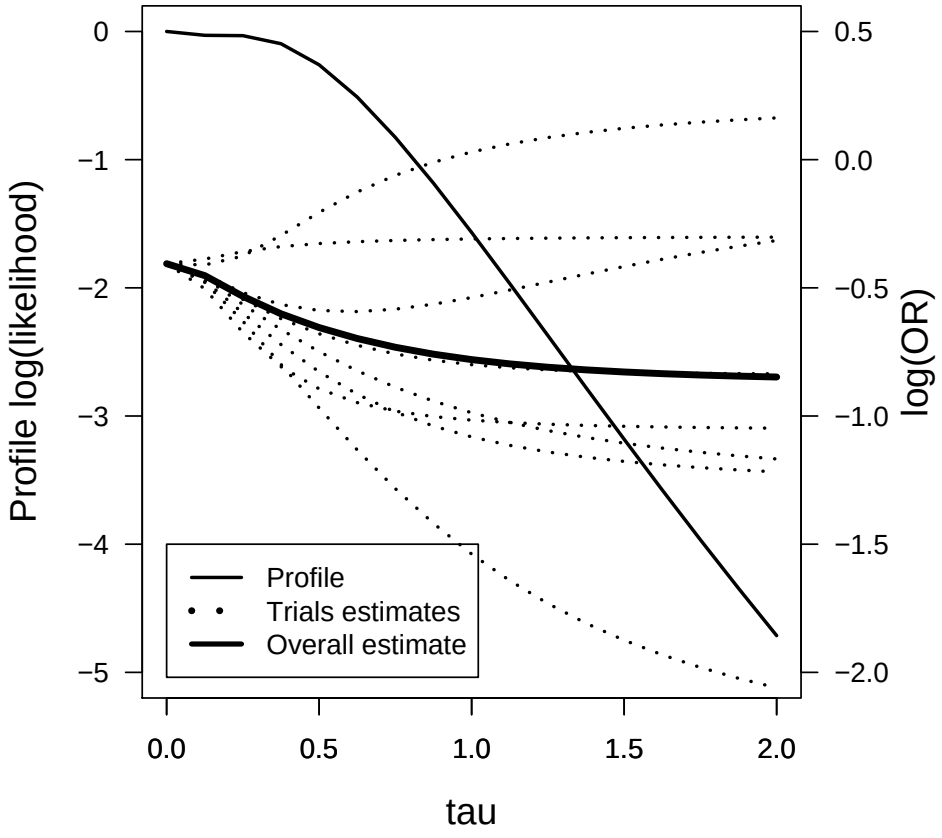

Figure 2.2: Profile log(likelihood) of τ, showing reasonable support for values of τ between 0 and 1. also shown are individual and overall estimates of treatment effects for different values of τ: although $\tau = 0$ is the maximum likelihood estimate, plausible values of τ have substantial impact on the estimated treatment effects.

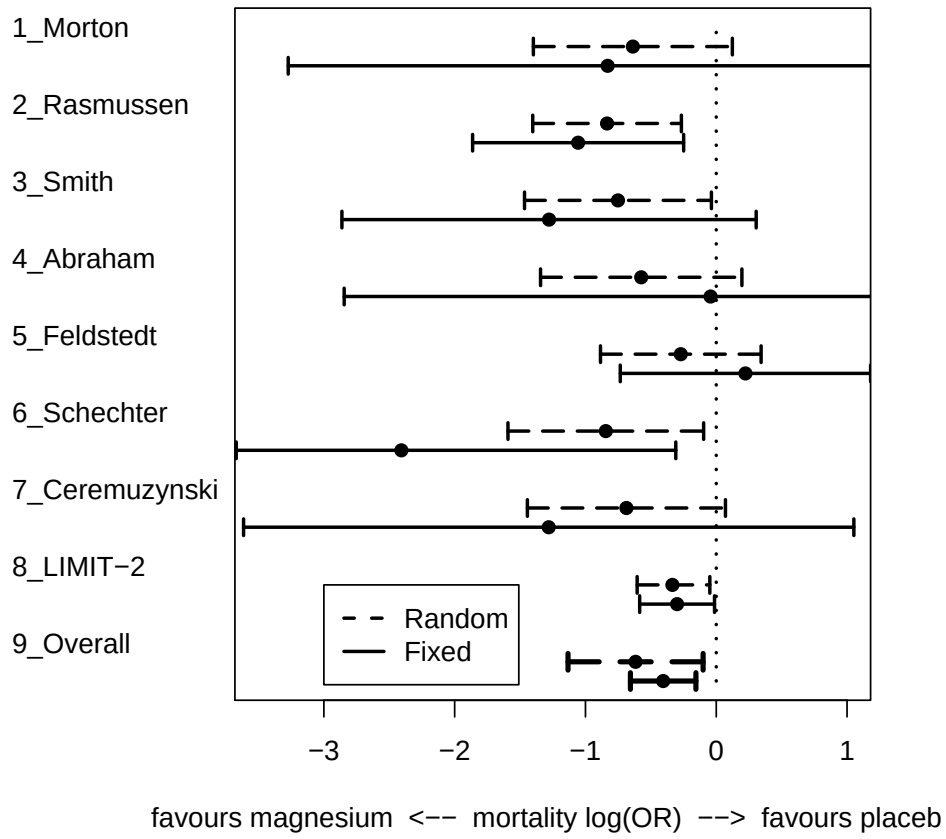

Figure 2.3: Fixed- (solid lines) and random-effects (dashed lines) meta-analysis of magnesium data assuming $\tau = 0.41$, leading to considerable shrinkage of the estimates towards a common value. The "Overall" figure is the pooled estimate from each analysis.

Bibliography

[1] Collins R, Peto R, Flather M and ISIS-4 Collaborative Group. (1995). ISIS-4 – a randomised factorial design assessing early orla captopril, oral mononitrate, and intravenous magnesium sulphate in 58,050 patients with suspected acute myocardial infarction. *Lancet* **345**, 669–685.

[2] DerSimonian R, Laird N. (1986). Meta-analysis in clinical trials. *Controlled Clinical Trials* **7**, 177–188.

[3] Higgins JP, Spiegelhalter DJ. (2002). Being sceptical about meta-analysis: a Bayesian perspective on magnesium trials in myocardial infarction. *International Journal of Epidemiology* **31**, 96–104.

[4] Spiegelhalter DJ, Abrams KR and Myles JP. (2004). *Bayesian Approaches to Clinical Trials and Health-Care Evaluation.* John Wiley and Sons, Ltd: Chichester.

[5] Sutton A, Abrams KR, Jones DR, Sheldon TA, Song F. (2000). *Methods for meta-analysis in medical research.* John Wiley and Sons, Ltd, Chichester.

[6] Teo KK, Yusuf S, Collins R, Held PH and Peto R. (1991). Effects of intravenous magnnesium in suspected acute myocardial infarction: overview of randomised trials. *British Medical Journal* **303**, 1499–1503.

[7] Whitehead A. (2002). *Meta-analysis of controlled clinical trials.* John Wiley and Sons, Ltd, Chichester.

[8] Woods KL, Fletcher S, Roffe C, Haider Y. (1992). Intravenous magnesium sulphate in suspected acute myocardial infarction: results of the Second Leicester Intravenous Intervention Trial (LIMIT-2). *Lancet* **339**, 1553-1558.

[9] Yusuf S. (1997). Meta-analysis of randomised trials: Looking back and looking again. *Controlled Clinical Trials* **18**, 594-601.

[10] Yusuf S, Teo K, Woods K. (1993). Intravenous magnesium in acute myocardial infarction: an effective, safe, simple and inexpensive treatment. *Circulation* **87**, 2043-2046.

2.9 Linear mixed models of fetal growth

Open the BUGS Compound document `fetal.odc` and have a look at the code, which implements the linear mixed model discussed in the lecture relating fetal head circumference to gestational age (where the assumed linear relationship is between the square root of head circumference and a fractional polynomial (in this case a quadratic) transformation of gestational age). Start by identifying the names of the nodes corresponding to the following quantities in the model:

Y_{ij}	Transformed measured of head circumference	`Y[ij]`
X_{ij}	Transformed measured of gestational age	`X[ij]`
β_0	Fixed effect intercept	`mu.beta[1]`
β_1	Fixed effect gradient	`mu.beta[2]`
$\beta_0 + u_{0i}$	Random effect intercept for subject i	`sub.beta[i,1]`
$\beta_1 + u_{1i}$	Random effect gradient for subject i	`sub.beta[i,2]`
u_{0i}	Random effect intercept deviation for subject i	`u[i,1]`
u_{1i}	Random effect intercept deviation for subject i	`u[i,1]`
$(\beta_0 + u_{0i})$ $+(\beta_1 + u_{1i})X_{ij}$	Conditional mean of Y_{ij} given β_0, β_1, u_{0i}, u_{1i}, X_{ij}	`mu[ij]`
Σ	Variance-covariance matrix for the random effects	`Sigma2.beta`
Ω	Inverse of Σ with Wishart prior	`Omega.beta`
σ_e	residual or error standard deviation	`sigma.e`
σ_e^2	residual or error variance	`sigma2.e`

Table 2.3: Names of parameters and nodes in the BUGS code

1. Compile the BUGS model in the file `fetal.odc` and run for 20,000 iterations, discarding the first 10,000 as a burn-in. Use the initial values for the parameters provided in the `fetal.odc` file. Generate summary statistics for the following nodes:

   - The fixed effect intercept and gradient `mu.beta[1]` and `mu.beta[2]` respectively.
   - The (symmetric) variance-covariance matrix `Sigma2.beta[]` for the random effects (where entry `[1,1]` is the variance of the random intercept, entry `[2,2]` is the variance of the gradient and entry `[1,2]` is the covariance between the random effect intercept and gradient.
   - The residual variance `sigma2.e`.

   You might like to verify your results (and the choice of starting values for the parameters provided in the file `fetal.odc`) using the `lme` routine in `R` to fit the same linear mixed model. Use the syntax

   ```
   linmod <- lme(SQRTHC ~ 1 + TGA, data = hc, random = ~ 1 + TGA |
   ID)
   ```

   to create a model object named `linmod`. In this syntax the dataframe is named `hc` and contains variables for the square root of head circumference `SQRTHC` (same as `Y` in the `WinBUGS` model), transformed gestational age `TGA` (same as `X` in the `WinBUGS` model) and subject identifier `ID`.

2. (a) Use the posterior means of the entries of the random effects variance-covariance matrix `Sigma2.beta[]` (for a description of the entries see question 1 above) to estimate the correlation of the random effect intercept and gradient. What is your interpretation of this estimate?

   (b) Create a node called `rancorr` and set it equal to the correlation of the random effect intercept and gradient based on the entry of the random effects variance-covariance matrix `Sigma2.beta`. That is, set `rancorr <- Sigma2.beta[1,2]/(sqrt(Sigma2.beta[1,1])* (sqrt(Sigma2.beta[2,2])))`. Recompile and runs the `BUGS` model, and output summary statistics for this node. Is the posterior mean of `rancorr` similar to the point estimate generated in part (a) of the question? Is there much support for values of `rancorr` higher or lower than its posterior mean?

   (c) It is possible to alter the parametrisation of the model to reduce the correlation between the random intercept u_{0i} and the random gradient u_{1i}. Consider "centering" the tranformed gestational age X_{ij} by subtracting a fixed constant c, redefining the transformed gestational age as $X_{ij}^{'} = X_{ij} - c$. Re-write the linear mixed model in terms of $X_{ij}^{'}$ (defining new random intercepts and gradients $u_{0i}^{'}$ and $u_{1i}^{'}$ which are functions of the original u_{0i} and u_{1i} and the constant c). What value of c will ensure that the correlation between $u_{0i}^{'}$ and $u_{1i}^{'}$ is zero? Can you alter the `WinBUGS` code to demonstrate this empirically?

3. One of the features of `BUGS` is the ability to generate predictive distributions for unobserved quantities by specifying these quantities as nodes in the graphical model used by `BUGS` to generate the simulations. Here we compare the unconditional predictive distribution of (transformed) head circumference at 38 weeks (transformed) gestational age with the corresponding *conditional* distribution given the value of the same fetal dimension at 18 weeks gestational age.

   Details for the five observations made on fetus `id = 5` are as follows:

id	ga	hc	Y	X
5	18.43	125	11.18	14.47
5	24.43	232	15.23	17.47
5	28.43	297	17.23	19.00
5	34.43	323	17.97	20.60
5	38.43	338	18.38	21.20

   For fetus `id = 5`, we can capture the conditional distribution of transformed head circumference at the final gestational age 38.43 weeks given the observed measurement at the first gestational age of 18.43 weeks by creating a new `id = 708` (we have observed data for 707 fetuses) with identical data for the first gestational age but no observed head circumferences measurements at the final gestational age:

id	ga	hc	Y	X
708	18.43	125	11.18	14.47
708	38.43	NA	NA	21.20

Note that these are observations 3098 and 3099 in our expanded data array, since we have original observations on 3097 occasions. `BUGS` will generate values for the "missing" node `Y[3099]` conditional on the observed for observation 3098 which we have indicated are from the same fetus since they share the same `id` number (`id = 708`). We should also add observation 3100 for a second new fetus to generate the *unconditional* distribution of (transformed) head circumference at 38.43 weeks gestational age:

```
id  ga     hc     Y       X

709   38.43  NA   NA      21.20
```

(a) Extend the data array as described above, change the maximum index for `i` to 709 and for `j` to 3100, and then recompile and rerun the `BUGS` model. Monitor node `Y[3099]` and `Y[3100]` which contain the conditional and unconditional transformed head circumference. Comment on the difference in posterior means - is this large in comparison to the posterior standard deviation of these two nodes? What is the appropriate interpretation of the conditional posterior mean (for fetus `id = 5`) being larger or smaller than the unconditional posterior mean?

(b) In this case we have the observed value of the (transformed) head circumference at gestational age 38.43 weeks for fetus `id = 5`. Calculate a conditional z-score for this observed value by subtracting the conditional mean (posterior mean of `Y[3099]`) at 38.43 weeks and dividing by the corresponding standard deviation. What is your interpretation of this z-score? Calculate the corresponding z-score using the unconditional values (from node `Y[3100]`) and compare this to the conditional z-score - does it make sense?

(c) It is straightforward to calculate the unconditional mean and standard deviation for transformed head circumference at gestational age 38.43 weeks (transformed gestational age 21.20) using the linear regression equation and the formula for the variance of a single observation, which is a quadratic function of transformed gestational age involving the variance-covariance parameters of the random effects and the error variance. Use the posterior means of the components of the vector `mu.beta[]` and the matrix `Sigma2.beta[]` to calculate explicitly the mean and standard deviation of the transformed head circumference at 38.43 weeks gestational age. How close are the calculated values to the posterior mean and standard deviation of `Y[3100]`?

2.10 Classical twin model in BUGS

Risk factors for mammographic density using twin data

Women with extensive dense breast tissue determined by mammography are known to be at higher risk of breast cancer than women of the same age with lower breast density. We will use data from a study of female monozygous (MZ) and dizygous (DZ) twin-pairs in Australia and North America to analyse the within-pair correlation of breast density, adjusted for age and weight.

The BUGS file `mgram.odc` contains computing code for the series of models outlined in the questions below. The following table describes the variables in the dataframe (which is also available as the Microsoft Excel file `mgram.xls` and the R data frame `mgram`):

`pdens1`	Percent mammographic density twin 1
`pdens2`	Percent mammographic density twin 2
`weight1`	Weight (kg) twin 1
`weight2`	Weight (kg) twin 2
`mz`	Indicator of MZ pair (1 = MZ, 0 = DZ)
`dz`	Indicator of DZ pair (1 = DZ, 0 = MZ)
`agemgram1`	Age in years of twin 1 at mammogram
`agemgram2`	Age in years of twin 2 at mammogram
`study`	Location indicator (1 = Australia, 0 = North America)

Table 2.4: Names of variables in the BUGS data from the mammographic density example.

1. Recall the basic hierarchical model for paired data described in lectures:

$$\begin{aligned} y_{i1} &= a_i + \varepsilon_{i1} \\ y_{i2} &= a_i + \varepsilon_{i2} \end{aligned}$$

where

$$\begin{aligned} \varepsilon_{ij} &\sim \mathrm{N}(0, \sigma_e^2) \\ \mathrm{cov}(\varepsilon_{i1}, \varepsilon_{i2}) &= 0 \\ a_i &\sim \mathrm{N}(\mu, \sigma_a^2) \end{aligned}$$

   (a) In order to compile the corresponding BUGS model and set it running, we need starting values for the parameters μ, σ_a^2 and σ_e^2. Note that $\frac{1}{2}(\mathrm{var}(y_{i1}) + \mathrm{var}(y_{i2})) = \sigma_a^2 + \sigma_e^2$ and that $\frac{1}{2}(\mathrm{var}(y_{i1} - y_{i2})) = \sigma_e^2$. Calculate the empirical values of $\mathrm{var}(y_{i1})$, $\mathrm{var}(y_{i2})$ and $\mathrm{var}(y_{i1} - y_{i2})$, and use these in a "methods of moments" calculation to produce estimates of σ_a^2 and σ_e^2 and hence generate starting values for σ_a and σ_e (since we are placing noninformative prior distributions on the standard deviation rather than the variance). You can use the sample mean of either y_{i1} or y_{i2} as the starting value for μ.

   (b) Compile the BUGS code and generate 1,000 iterations for summary after a burn-in of 1,000 iterations. What are the posterior means and standard deviations of μ, σ_a^2 and σ_e^2?

   (c) Use the posterior means of σ_a^2 and σ_e^2 to estimate the within-pair correlation of y_{i1} and y_{i2}.

2. In question 1 we assumed a constant within-pair correlation for y_{i1} and y_{i2}, in particular that this correlation is the same for MZ and DZ pairs. If the outcome is influenced by genetic factors then this is unlikely to be a satisfactory assumption. Use the second set of `BUGS` code to compile a model that uses an additional parameter `rho` ($\rho_{DZ:MZ}$ from lectures) to represent the ratio of $\text{cov}(y_{i1}, y_{i2})$ in DZ and MZ pairs. We assign `rho` a starting value of 0.5, and use the starting values from question 1 for the remaining parameters.

   (a) Generate a table of posterior summary statistics for the four parameters μ, σ_a^2, σ_e^2 and $\rho_{DZ:MZ}$.

   (b) How have the posterior means of σ_a^2 and σ_e^2 changed now that DZ and MZ pairs can have distinct within-pair correlations? How should this change be interpreted?

   (c) Does the posterior mean value for $\rho_{DZ:MZ}$ suggest that there are genetic factors determining the value of mammographic density? Is the posterior estimate of $\rho_{DZ:MZ}$ consistent with an additive genetic model?

3. Previous research has established that age-adjusted mammographic density is a risk factor for breast cancer. We can include this adjustment in our `BUGS` model from the previous question by using an extra parameter (node) `b.age` in our model, and including the terms `b.age*agemgram1` and `b.age*agemgram2` in the mean model for mammographic density `pdens1` and `pdens2` in twins 1 and 2 respectively.

   (a) Generate a starting value for `b.age` by regressing percent mammographic density on age at mammogram in `R` using data from either twin 1 or twin 2 (or both if you're motivated to concatenate the data vectors).

   (b) Use the starting value in part (a) to compile and run the `BUGS` model with adjustment for age, and produce a summary table of the posterior distributions for the parameters μ, σ_a^2, σ_e^2, $\rho_{DZ:MZ}$ and β_{age} = `b.age`. Is there evidence for a linear relationship between mammographic density and age at mammogram?

   (c) Has the adjustment for age changed the posterior mean of $\rho_{DZ:MZ}$? Is the current posterior mean for $\rho_{DZ:MZ}$ consistent with an additive genetic model for mammographic density?

4. Our final adjustment is to include `weight` in our regression model for mammographic density which also includes age at mammogram. We include this variable in our `BUGS` model in the same way as we did in the previous question for the `agemgram` variable: Use an extra parameter (node) `b.wgt` in the model, and including the terms `b.wgt*weight1` and `b.wgt*weight2` in the mean model for mammographic density `pdens1` and `pdens2` in twins 1 and 2 respectively.

   (a) Generate a starting value for `b.wgt` by regressing percent mammographic density on weight and age at mammogram in `R` using data from either twin 1 or twin 2 (or both if you're motivated to concatenate the data vectors).

   (b) Use the starting value in part (a) to compile and run the `BUGS` model with adjustment for weight, and produce a summary table of the posterior distributions for the parameters μ, σ_a^2, σ_e^2, $\rho_{DZ:MZ}$ and β_{age} = `b.age` and β_{weight} = `b.wgt`. Is there evidence for a linear relationship between mammographic density and weight adjusted for age at mammogram?

(c) Has the adjustment for age changed the posterior mean of $\rho_{DZ:MZ}$? Is the current posterior mean for $\rho_{DZ:MZ}$ consistent with an additive genetic model for mammographic density?

2.11 Using the DIC in model comparison

In this exercise we work through an example that demonstrates the importance of defining the *focus* (i.e. set of parameters) of a model comparison. This example is courtesy of Bob O'Hara and appears on his website
deepthoughtsandsilliness.blogspot.com/2007/12/focus-on-dic.html

Suppose there are $m = 10$ groups of data (indexed by $i = 1, \ldots, m$) each with $n = 50$ observations (indexed by $j = 1, \ldots, n$) that have been generated from the two-level normal-normal hierarchical model:

$$\begin{aligned} Y_{ij}|\theta_i &\sim \mathrm{N}(\theta_i, \sigma^2) \\ \theta_i|\mu_i, \tau &\sim \mathrm{N}(\mu_i, \tau^2) \end{aligned}$$

We consider two models for the group-specific mean parameter μ_i:

$$\begin{aligned} \text{Model 1: } \mu_i &= \mu + \beta(i - 5.5) \\ \text{Model 2: } \mu_i &= \mu \end{aligned}$$

The first model has a covariate (equal to the identity number of the group) but the second has none.

1. Use the R code to simulate data Y_{ij} according to the two models above (call them Data 1 and Data 2 respectively), and plot the data in each group along with the observed group specific mean.

You should see from the plot that the effect of the covariate is clear, so the DIC should be able to pick it up.

2. Fit each of the models to each of the two simulated data sets, using the R code to run the `WinBUGS` models through R. Extract the DIC from each model and compare them. Is the DIC lower for the model that includes the covariate when fitted to the data simulated using the group-specific covariate, compared to fitting the model without the covariate?

You should have found that in both cases the DIC is the same (for most simulations the difference is no higher than the third decimal place). But for the data simulated with a group-specific covariate (Data 1), Model 1 should be better, as suggested by the earlier plots. So what's going on? We can get a clue from plotting the posteriors of μ_i for each of the groups, from the two models.

3. Use the R code to plot the group-specific means for both Data 1 and Data2, with errors bars (i.e. $\pm$ 1 posterior standard deviation), along with the 1:1 identity line.

Obviously the models are predicting the same means for the groups, and hence we will get the same deviance (recall that we are talking about the plug-in deviance here which depends only on the posterior means of the parameters on which we are focussing). We can see why this is happening from the between-group or group-level standard deviations.

4. Use the output from the `WinBUGS` run to calculate the posterior mean and standard deviation of the between-group or group-level standard deviation parameter τ for both Model 1 and Model 2 applied to Data 1 and Data 2.

You should have found that for the data where there is a trend (Data 1), but none is fitted, the posterior mean of τ is much larger. The lack of the linear trend is compensated by the increase in variance. The difference is not in the model for θ at all, but occurs higher in the hierarchy at the level of the hyperparameter μ where the effect of the group-specific covariate is incorporated into the model.

This is obvious from looking at the models. In order for it to be reflected in a comparison of the DIC between models, we need to change the focus, from θ to μ and β. This then means calculating the *marginal* deviance, marginalising over θ, that is, looking at $p(\mathbf{Y}|\mu, \tau)$ after integrating $p(\mathbf{Y}|\theta)$ over $p(\theta|\mu, \tau)$. This can be done analytically, after which we find that the deviance can be calculated because we know the distribution of the group-specific sample mean $\overline{Y}_{i.} = \sum_{j=1}^{n} Y_{ij}/n$, which is

$$\overline{Y}_{i.} \sim \mathrm{N}(\mu_i, \sigma^2/n + \tau^2). \tag{2.8}$$

5. Recalculate the DIC for each dataset and each model using the functions provided in the R code.

The results should now make more sense. For the data with a covariate effect for the mean model (Data 1), the DIC massively favours the correct model. Without the effect in the data, the DIC is pretty similar for the two models. In both cases, also note that p_D is larger by 1 for the model with 1 extra parameter, as expected.

What lessons can we draw from this? Firstly, that DIC is not an automatic panacea - it must be focussed on the right part of the model. If the focus is not at the level immediately above the data (i.e. θ here), then you can't use the DIC given by `BUGS`. In this example it is more difficult to get at the correctly focussed DIC (in fact you have to calculate it manually yourself, or at least use Bob O'Hara's R function to do so). For more complex models this might be awkward, since if there are no analytical results, then the parameters to be integrated out have to be simulated, for example by Markov chain Monte Carlo.

Some comments from Martyn Plummer:

This example encourages you to think about what DIC is trying to do. It's not about finding the "true" model - both models are true in fact - it's about accurately predicting dropped observations.

In the simulated data, there are 50 observations in each group. If you drop one observation and then tried to predict it, you already have plenty information from the other 49 observations in the same group that share the same mean, and you have 489 degrees of freedom to estimate the variance. The group-level covariate really doesn't add much to your ability to make that prediction.

Changing the focus to the group level, you are dropping a whole group and then trying to predict the 50 observations in it. In this case, the group-level covariate is very useful. Here DIC parts company with the penalized plug-in likelihood since we have around 3 effective parameters and only 10 independent observations! You'd most likely be better off using the "corrected" DIC proposed in the Discussion of Plummer (2008). Although the calculations haven't been done explicitly, the substantive conclusions must surely be the same.

2.12 Measurement comparison in oximetry

A common problem in medical statistics is assessing the extent to which a new technique for measuring a biological quantity gives results that agree with a more established method of measurement. An important example arises in *oximetry* which is the measurement of the saturation or concentration of oxygen in the blood. Patients who are critically ill are unable to send enough oxygen into the bloodstream and the level of oxygen saturation is monitored as an indicator of the severity of the patient's condition. The traditional method of measurement uses a sample of blood on which a chemical analysis is performed to determine the level of various gases in the blood ("co-oximetry"). A much more convenient, newer, method uses a device called a pulse oximetry, which relies on a small sensor placed on a finger or toe to measure oxygen saturation by measuring the reflectance of light through the blood vessels.

A study was done at the Royal Children's Hospital in Melbourne to examine the agreement between pulse oximetry and co-oximetry in small babies, many of whom were especially sick and therefore had oxygen saturation levels lower than those usually available to test the accuracy of pulse oximetry. The data file (`ox.dat` or `ox.csv`) contains 4 variables on a total of 61 babies.

Variable name	Description
`item`	Subject identifier
`repl`	Number of sample (replicate)
`co`	Oxygen saturation (%) by co-oximetry
`pulse`	Oxygen saturation (%) by pulse oximetry

There were 61 babies in the study, each contributed up to 3 samples, but in a few cases only one or two measurements were available; in total there are 177 observations.

1. To begin with we model the differences d_{ir} for the r^{th} sample (replicate) ($r = 1, 2, 3$) on the i^{th} infant as normally distributed: $d_{ir} \sim \mathcal{N}(\delta, \sigma^2)$. Note that in this case we ignore the clustering within subjects and analyze the data as 177 independent observations of the differences.

   The simplest model is one with a mean difference between methods (the average difference) and a standard deviation of this differences:

   $$d_{ir} \sim \mathcal{N}(\delta, \sigma^2)$$

   (a) Fit the model using `lm`. What is mean difference, and what is the standard deviation of this?

   (b) If we use uninformative priors for σ^2, and δ *i.e.* $p(\sigma^2) \propto \sigma^{-2}$ and $p(\delta) \propto 1$. What is the posterior distribution $p(\sigma^2|d_{ir})$ under the above assumptions? Calculate a 95% posterior interval for σ^2.
   Hint: See section 3.2 of *Bayesian Data Analysis*.

   (c) What is the posterior distribution $p(\delta|d_{ir})$? Calculate a 95% posterior interval for δ.

   (d) Define this model in `BUGS` with the uninformative priors, and run it. The data are available as `ox.dat` or `ox.csv` from the course homepage. These file can be read into R by `read.table` and `raed.csv`, respectively.

   How do the results agree with what you found above?

   (e) Derive 95% posterior intervals for both δ and σ^2 using the `BUGS` output.

(f) The 95% *range of agreement* is defined as $\delta \pm 2\sigma$, a prediction interval for a future difference between methods. Now introduce these limits as nodes `agree.lo` and `agree.hi` in the `BUGS` code, and re-run it. Is this necessary?

(g) Suppose we have a prior distribution for the mean δ that is $\mathrm{N}(0, 1.5^2)$ (*i.e.*, we would be surprised if a device like this was systematically biased by more than 3%) and continue to use the standard noninformative prior on σ^2.

Compare the posterior in this case with the previously obtained. Does the informative prior distribution for δ have any impact on the posterior distribution for σ^2?

2. So far we have regarded the three observations on each infant as independent. But since they are from the same child, it is likely that they are correlated. Moreover, as the measurements are taken at three different time (in *pairs* of co and pulse) the measurements taken at the same time are likely to be similar.

We therefore first introduce a *subject-specific* effect μ_i shared by all measurements on the i^{th} infant:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + \delta + e_{\text{pulse},ir} \end{aligned}$$

where $e_{mij} \sim \mathrm{N}(0, \sigma_m^2)$, $m = \text{co}, \text{pulse}$. Note that the error terms for the two methods are different as it would rather daft to assume that me measurement error were the same for the two.

(a) What is distribution of $d_{ir} = y_{\text{co},ir} - y_{\text{pulse},ir}$ under this model?

(b) Amend the `BUGS` code to accommodate this model A suitable noninformative prior distribution for σ_m is a uniform distribution on $[0, K]$, where K is a suitably large number (recall that the posterior will also have finite support in this case. Run the model.

(c) Generate and display posterior summaries of the estimated standard deviations. I there strong evidence that one of these residual standard deviation is bigger than the other? Does extending the model in this way influence our inference about δ?

3. The previous model allows separate residual variances for the two measurement errors, but the model still assumes exchangeability of replicates *within* methods. But the replicates are *linked*; they are taken at three different timepoints, so they measure something potentially slightly different — there may be a time-to-time variability within the infant which is common for the two methods, so we try to incorporate this via a random effect a_{ir} with variance ω^2:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + a_{ir} + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + \delta + a_{ir} + e_{\text{pulse},ir} \end{aligned}$$

(a) Modify your `BUGS` code to accomodate this new variance component. You wil also need to supply the replicate number `repl` in the data and in the code you will need the refer to the singe random effects by using nested indexing as `a[i,meth[i]]`.

(b) Make a traceplot for the resulting `mcmc.list`. What is your conclusion — has the chains converged?

(c) Make a pairwise scatter plot of the parameters in the model. Use `as.matrix` to get a matrix of the posterior samples that you can stuff into `pairs`. What is your conclusion?

(d) The model can also be fitted by conventional methods, in this case we resort to `lme`. For this you must first stack the data, and then invoke the arcane syntax of `lme`:

```
> oxl <- data.frame( y = c(oxw$co,oxw$pulse),
+                  repl = factor( rep(oxw$repl,2) ) ,
+                    id = factor( rep(oxw$item,2) ),
+                  meth = factor( rep(c("co","pulse"),each=177) ) )
> library( nlme )
> m1 <- lme( y ~ meth + id,
+                 random = list( id = pdIdent( ~ repl-1 ) ),
+                 weights = varIdent( form = ~1 | meth ),
+                 data = oxl,
+                 control = lmeControl(returnObject=TRUE) )
> m1
```

4. The difference in means between the two methods of measurement may not be the same for all levels of oxygen saturation. The simplest way to allow for this is to introduce a linear relationship between the means:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \alpha + \beta\mu_i + e_{\text{pulse},ir} \end{aligned}$$

Note that for $\beta = 1$ this is the earlier model.

(a) Extend the `BUGS` model to include the linear relationship between means. Use a noninformative prior distribution for α, but constrain β to lie between 0 and 2 (it must be positive and the "null" value for constant mean difference between methods is $\beta = 1$).

Generate and display posterior summary statistics for α and β. Is there strong evidence against the null hypothesis that $\beta = 1$?

(b) In the previous question we used co-oximetry as the reference method, with μ_i as the mean for $y_{\text{co},ir}$. We might as well have chosen pulse-oximetry as the reference method and re-expressed the model as

$$\begin{aligned} y_{\text{co},ir} &= \alpha^\star + \beta^\star\mu_i + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + e_{\text{pulse},ir} \end{aligned}$$

Change the `BUGS` code to use pulse-oximetry as the reference method, using the same prior distributions for $\alpha^\star$ and $\beta^\star$ as were used for α, β and μ_i. Provide summaries of the posterior distribution of α^* and β^*.

(c) What are the relations between (α, β) and $(\alpha^\star, \beta^\star)$? Check whether the relation holds between the results from the two previous fits of the model.

(d) Compare the results for α and β frrom the two previous models with the results of regressing `co` on `pulse` and vice-versa.

5. In order to get the model right we must reformulate it so that it is symmetric in the two methods:

$$\begin{aligned} y_{\text{co},ir} &= \alpha_{\text{co}} + \beta_{\text{co}}(\mu_i + a_{ir}) + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \alpha_{\text{pulse}} + \beta_{\text{pulse}}(\mu_i + a_{ir}) + e_{\text{pulse},ir} \end{aligned}$$

Formally this is the same model as the two above, but the conversion it is formulatred symmetrically in the parameters. However it is also over-parametrized.

(a) How are the means of the two methods related in this model?

(b) (*Can be omitted*) What happens to the αs and βs if the μs are linearly transformed? Try to reparaetrize $\mu_i = a + b\xi_i$ and express the model in the same form using the ξ_is. How does this influence the way the means of the two methods are related?

(c) Modify your `BUGS` program to fit this model and run it.

(d) Check the mixing of the chains using `xyplot` and inspect the two-dimensional posteriors using `pairs(as.matrix())` on the resulting `mcmc.list` object.

(e) How do the posterior means relate to the two sets of regression parameters previously?

(f) Check the convergence of the chains graphically and numerically

(g) Formulate a conclusion regarding the two methods of measurement.

Chapter 3

Solutions

3.1 Bayesian inference in the binomial distribution

1. In the discrete case we just set up a vector if the same length as the prior — we know that the likelihood and posterior only are defined in the points where the prior is positive.

   (a) In Rwe just do the computations according to the rules, and the print the vector side by side corresponding to the table in the exercise:

   ```
   > theta <- c(2,4,6,8)/10
   > prior <- c(1,1,1,1)/4
   > x <- 1
   > n <- 1
   > like  <- dbinom( x, n, theta )
   > like.pr <- prior * like
   > post  <- like.pr / sum( like.pr )
   > round( cbind( theta, prior, like, like.pr, post ), 3 )
   ```

   ```
        theta prior like like.pr post
   [1,]   0.2  0.25  0.2    0.05  0.1
   [2,]   0.4  0.25  0.4    0.10  0.2
   [3,]   0.6  0.25  0.6    0.15  0.3
   [4,]   0.8  0.25  0.8    0.20  0.4
   ```

   Not surprising, the posterior is proportional to the likelihood when we use a uniform prior as in this case. And since the likelihood is maximal for $theta = 1$, we get the maximal posterior probability for $\theta = 0.8$, the largest possible value.

   (b) If we had 20 trials and 15 successes we just change the value of `x` and `n` in the code:

   ```
   > theta <- c(2,4,6,8)/10
   > prior <- c(1,1,1,1)/4
   > x <- 15
   > n <- 20
   > like  <- dbinom( x, n, theta )
   > like.pr <- prior * like
   > post  <- like.pr / sum( like.pr )
   > round( cbind( theta, prior, like, like.pr, post ), 3 )
   ```

   ```
        theta prior  like like.pr  post
   [1,]   0.2  0.25 0.000   0.000 0.000
   [2,]   0.4  0.25 0.001   0.000 0.005
   [3,]   0.6  0.25 0.075   0.019 0.298
   [4,]   0.8  0.25 0.175   0.044 0.697
   ```

   We see the same patterns as before. The 0 posterior for $\theta = 0.2$ is not an exact 0; it is just a consequence of rounding:

```
> round( cbind( theta, prior, like, like.pr, post ), 17 )
```

```
     theta prior         like      like.pr         post
[1,]   0.2  0.25 1.664729e-07 4.161823e-08 6.645594e-07
[2,]   0.4  0.25 1.294494e-03 3.236234e-04 5.167614e-03
[3,]   0.6  0.25 7.464702e-02 1.866175e-02 2.979907e-01
[4,]   0.8  0.25 1.745595e-01 4.363988e-02 6.968411e-01
```

(c) If we expand the set of support points for the prior (and hence also for the posterior, should get an expansion of the support for the posterior too. But if $x \neq 0$, then the likelihood at $\theta = 0$ is 0, since this value of θ corresponds to a situation where an event never occurs. Likewise if $x \neq n$ the likelihood at $\theta = 1$ is 0, since this corresponds to a situation where an event always occurs.

If we have $x = 15$ and $n = 20$, the the likelihood at the two outer points will be the same and the posterior will also be the same (because the prior at the "remaining points" is the same as before, bar a constant:

```
> theta <- c(0,2,4,6,8,10)/10
> prior <- c(1,1,1,1,1,1)/6
> x <- 15
> n <- 20
> like  <- dbinom( x, n, theta )
> like.pr <- prior * like
> post  <- like.pr / sum( like.pr )
> round( cbind( theta, prior, like, like.pr, post ), 3 )
```

```
     theta prior  like like.pr  post
[1,]   0.0 0.167 0.000   0.000 0.000
[2,]   0.2 0.167 0.000   0.000 0.000
[3,]   0.4 0.167 0.001   0.000 0.005
[4,]   0.6 0.167 0.075   0.012 0.298
[5,]   0.8 0.167 0.175   0.029 0.697
[6,]   1.0 0.167 0.000   0.000 0.000
```

(d) If we only have a singe positive trial, we will however have a positive likelihood at $\theta = 1$:

```
> theta <- c(0,2,4,6,8,10)/10
> prior <- c(1,1,1,1,1,1)/6
> x <- 1
> n <- 1
> like  <- dbinom( x, n, theta )
> like.pr <- prior * like
> post  <- like.pr / sum( like.pr )
> round( cbind( theta, prior, like, like.pr, post ), 3 )
```

```
     theta prior like like.pr  post
[1,]   0.0 0.167  0.0   0.000 0.000
[2,]   0.2 0.167  0.2   0.033 0.067
[3,]   0.4 0.167  0.4   0.067 0.133
[4,]   0.6 0.167  0.6   0.100 0.200
[5,]   0.8 0.167  0.8   0.133 0.267
[6,]   1.0 0.167  1.0   0.167 0.333
```

2. In the continuous case we use the Beta-distribution, which is also available in R, so it is straightforward to do the same calculations as above. However we cannot just print the values of the prior, the likelihood and the posterior at the supported values, because the support is now the entire interval $[0, 1]$. Hence we compare by making graphs with an x-axis form 0 to 1.

   (a) The formulae given in the exercise immediately lend themselves to implementation in

R:

$$m = \frac{a}{a+b} \quad \Leftrightarrow \quad a = m(a+b)$$

$$s = \sqrt{\frac{m(1-m)}{a+b+1}} \quad \Leftrightarrow \quad a+b = \left(m(1-m)/s^2\right) - 1$$

The only thing we need to supply are the desired values of m and s:

```
> m <- 0.4
> s <- 0.1
> a.plus.b <- m*(1-m)/s^2 - 1
> a <- m * a.plus.b
> b <- a.plus.b - a
> c(m,s,a,b)

[1]  0.4  0.1  9.2 13.8
```

(b) For these values of a and b we can just use the Beta-density implemented in the `dbeta` function in R to plot the desired prior distribution function:

```
> # Points where we plot:
> p <- seq(from=0,to=1,length=100)
> # Graph of the prior
> plot( p, dbeta( p, a, b ), lwd=4, bty="n", type="l" )
```

(c) For an observation of $x = 15$ out of $n = 20$ we use the `dbinom` function with the probability `p` as the argument to plot the likelihood:

```
> x <- 15
> n <- 20
> plot( p, dbinom( x, n, p ), lwd=4, bty="n", type="l" )
```

(d) We know that the posterior is a Beta-distribution with parameters $a + x$ and $b + n - x$, so this is just as easily implemented in R:

```
> plot( p, dbeta( p, a+x, b+n-x ), lwd=4, bty="n", type="l" )
```

(e) In order to see how the three relate we collect the three plots in one frame:

```
> par( mfcol=c(3,1) )
> plot( p, dbeta( p, a, b ), lwd=4, bty="n", type="l" )
> plot( p, dbinom( x, n, p ), lwd=4, bty="n", type="l" )
> plot( p, dbeta( p, a+x, b+n-x ), lwd=4, bty="n", type="l" )
```

which is slightly primitive; a more beefed-up version would be:

```
> par( mfcol=c(3,1), mar=c(3,3,0,0) )
> plot( p, dbeta( p, a, b ), lwd=4, bty="n", type="l" )
> text( par("usr")[1], par("usr")[4], "\n  Prior", adj=c(0,1) )
> plot( p, dbinom( x, n, p ), lwd=4, bty="n", type="l" )
> text( par("usr")[1], par("usr")[4], "\n  Likelihood", adj=c(0,1) )
> plot( p, dbeta( p, a+x, b+n-x ), lwd=4, bty="n", type="l" )
> text( par("usr")[1], par("usr")[4], "\n  Posterior", adj=c(0,1) )
```

The results of these two approaches are shown side-by-side in figure 2e.

(f) In order to illustrate the effect of variations in the prior and the data we wrap the calculations, and the graphing of the three functions in an R-function. The `text`-function draws text on the plot so it is possible to trace the parameters in the various plots.

```
> Bayes.ill <-
+ function( m, s, x, n, ... )
+ {
+ p <- seq(0,1,,1000)
+ a.plus.b <- m*(1-m)/s^2 - 1
+ a <- m * a.plus.b
```

```
+ b <- a.plus.b - a
+ plot( p, dbeta( p, a, b ), lwd=4, bty="n", type="l", ... )
+ text( par("usr")[1], par("usr")[4],
+       paste("\n  Prior\n  m=", m, ",s=", s,
+                       "\n  a=", a,", b=", b), adj=c(0,1) )
+ plot( p, dbinom( x, n, p ), lwd=4, bty="n", type="l", ... )
+ text( par("usr")[1], par("usr")[4],
+       paste("\n  Likelihood\n  n=", n,", x=",x), adj=c(0,1) )
+ plot( p, dbeta( p, a+x, b+n-x ), lwd=4, bty="n", type="l", ... )
+ text( par("usr")[1], par("usr")[4],
+       paste("\n  Posterior\n  Beta(", a+x, ",", b+n-x, ")"), adj=c(0,1) )
+ }
```

Note the argument "`...`" which allows us to pass extra parameters on the the plot statements. This function produces three plots, so when using it it will be convenient to set up a layout of plots using for example `par(mfcol=c(3,2))`, which gives a 3 by 2 matrix of graphs, filled column-wise. The `mar=` argument governs the whitespace around the single plot frames, and we use `col=gray(0.5)` to plot the curves in gray so that any text on top of them will be visible:

```
> par( mfcol=c(3,2), mar=c(2,4,0,0) )
> Bayes.ill( 0.4, 0.2, 15, 20, col=gray(0.5) )
> Bayes.ill( 0.4, 0.1, 15, 20, col=gray(0.5) )
```

```
> par( mfcol=c(3,2), mar=c(2,4,0,0) )
> Bayes.ill( 0.4, 0.2, 55, 100, col=gray(0.5) )
> Bayes.ill( 0.4, 0.1, 75, 100, col=gray(0.5) )
```

The results of these statements are shown in figure 2f.

3. The fraction of female births in most societies is around 48.7%. A reasonable prior would be one that is centered around 50% with a spead that is effectively så large that is will encompass even extreme deviations form the expected mean.

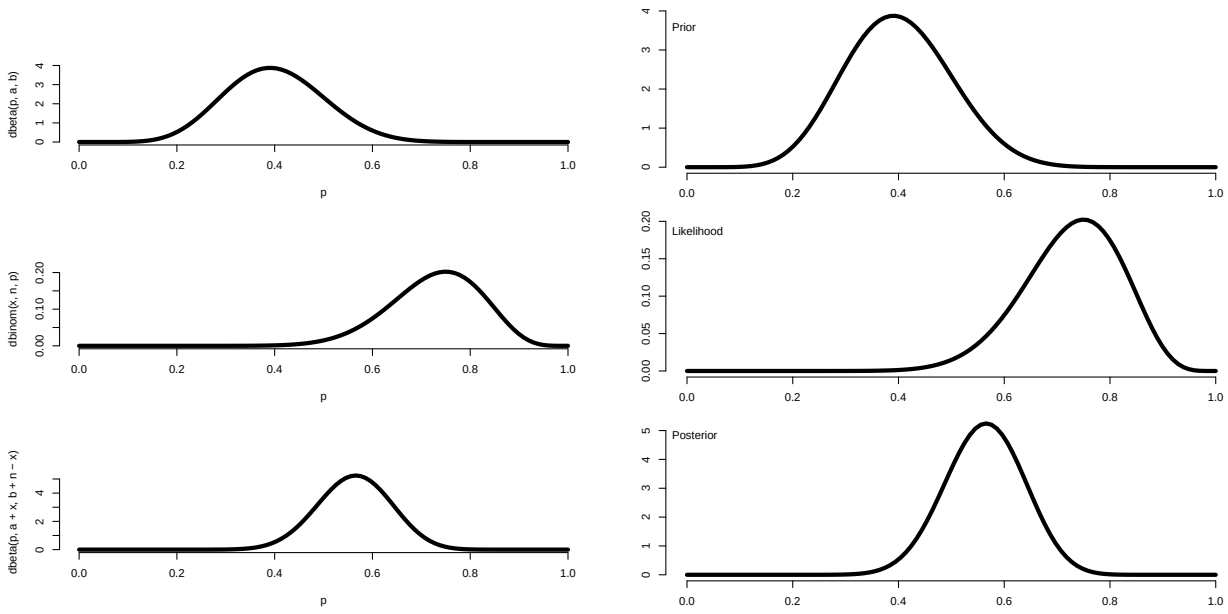

Figure 3.1: *Prior, likelihood and posterior for the binomial model. The right hand side is just the beefed-up version of the plot.*

(a) If we use a Beta(100,100) We can either make a numeric calculation for the probability that a Beta(100,100) variate is between 0.4 and 0.6:

```
> pbeta( 0.6, 100, 100 ) - pbeta( 0.4, 100, 100 )
[1] 0.9956798
```

or do a more brutal computation using a random sample:

```
> zz <- rbeta( 10000, 100, 100 )
> mean( zz<0.6 & zz>0.4 )
[1] 0.996
```

So we are indeed more than 95% certain that the true fraction of girls is between 40 and 60%!

(b) If we see 511 boys out of 1000 births, we can use the previous function to illustrate how the the prior, likelihood and posterior look in this problem. Note that we use the "`...`" argument to pass on a limitation of the x-axis:

```
> a <- b <- 100
> m <- a/(a+b)
> s <- sqrt(m*(1-m)/(a+b+1))
> par( mfcol=c(3,1), mar=c(4,2,0,0) )
> Bayes.ill( m, s, 511, 1000, xlim=c(0.4,0.6), xlab="% male births" )
> abline(v=0.5)
```

(c) The posterior probability that the fraction of female births i larger than 0.5 is the same the probability that the fraction of male births is < 0.5, is just a cumulative probability in the posterior distribution which is Beta(611,589):

```
> pbeta(0.5,611,589)
[1] 0.2626087
```

i.e. the prior and the data translates into a posterior probability of 26%. We see that the prior has a limited influence; a flat prior (Beta(1,1)) would have resulted in a posterior with parameters (511,489), and a smaller posterior probability:

```
> pbeta(0.5,512,490)
[1] 0.2434263
```

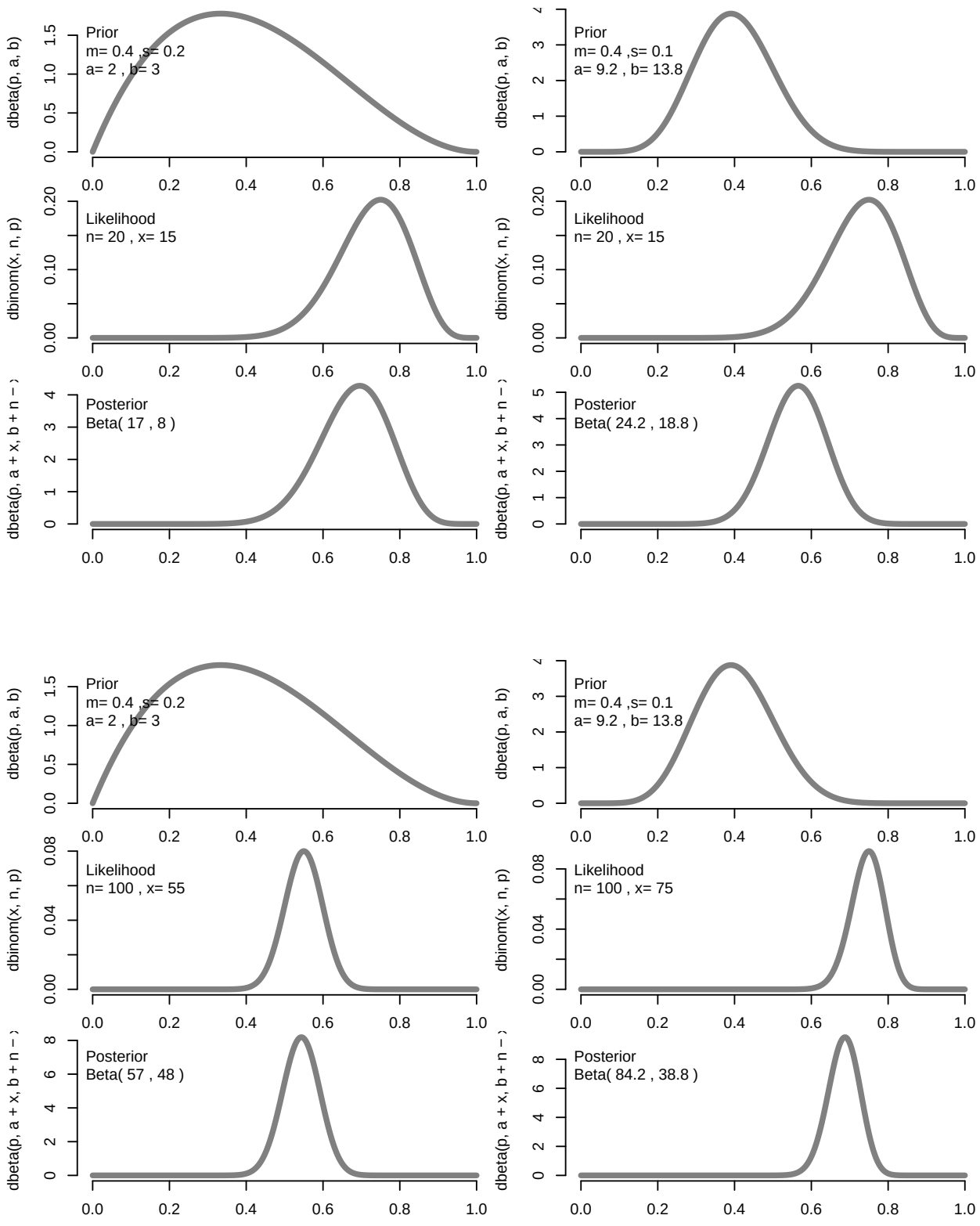

Figure 3.2: *Prior, likelihood and posterior for the binomial model for different combinations of prior information and data. Large amounts of data makes the likelihood the dominant factor; and a narrow prior (strong beliefs!) makes the prior the dominant factor.*

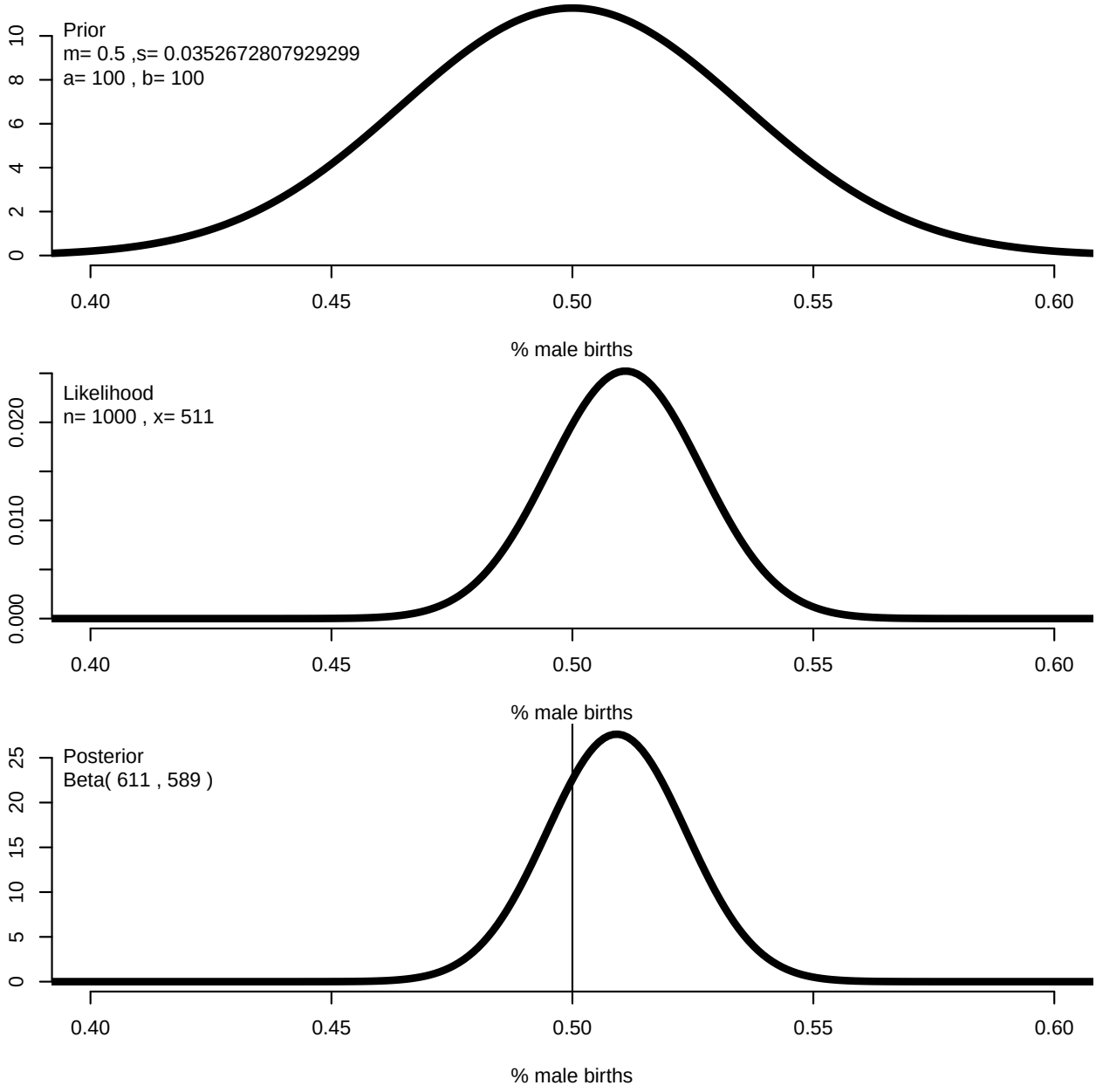

Figure 3.3: *Prior, likelihood and posterior for the binomial model for 511 births out of 100, using a Beta(100,100) prior. It is immediately apparent that the prior has very little influence on the posterior — all the information is in the likelihood, i.e. the data.*

3.2 Simple linear regression with BUGS

First we load all the required packages for this practical:

```
> library( R2WinBUGS )
> library( BRugs )
> library( Epi )
> # Get a function to convert bugs objects to mcmc.list objects
> source("../r/PDAwBuR.r")
```

1. Define and plot the bogus data and inspect the output from the linear regression analysis:

```
Call:
lm(formula = y ~ x)

Residuals:
       1        2        3        4        5        6
-0.09524  0.87619 -0.15238 -1.18095 -0.20952  0.76190

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.06667    0.78153   0.085  0.93612
x            1.02857    0.20068   5.125  0.00686

Residual standard error: 0.8395 on 4 degrees of freedom
Multiple R-squared: 0.8679,        Adjusted R-squared: 0.8348
F-statistic: 26.27 on 1 and 4 DF,  p-value: 0.00686
```

   The estimates of α and β are 0.067 and 1.029, and the estimate of σ is 0.840.

2. In order to use BUGS we set up the data, initial values (for three chains) and the list of parameters to monitor:

```
> reg.dat <- list( x=x, y=y, I=6 )
> reg.ini <- list( list( alpha=0.05, beta=1.0, sigma=0.9 ),
+                  list( alpha=0.04, beta=1.1, sigma=1.0 ),
+                  list( alpha=0.06, beta=0.9, sigma=1.1 ) )
> reg.par <- c("alpha","beta","sigma" )
```

   Finally we need to specify the model in BUGS code, using the names we specified for the data in `reg.dat`.

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          y[i] ~ dnorm(mu[i],tau)
+          mu[i] <- alpha + beta*x[i]
+          }
+       alpha ~ dnorm(0, 1.0E-6)
+       beta  ~ dnorm(0, 1.0E-6)
+       sigma ~ dunif(0,100)
+       tau <- 1/pow(sigma,2)
+       }",
+       file="reg.bug" )
```

   With these specifications we can now use `bugs()` to run the MCMC:

```
> reg.res <- bugs( data = reg.dat,
+                  inits = reg.ini,
+                  param = reg.par,
```

```
+                 model = "reg.bug",
+               n.chains = 3,
+                 n.iter = 20000,
+               n.burnin = 10000,
+                 n.thin = 5,
+                program = "openbugs",
+                clearWD = TRUE )

Initializing chain 1: Initializing chain 2: Initializing chain 3:

> reg.res <-  mcmc.list.bugs( reg.res )
```

The summary of the posterior distributions of the parameters can now be obtained by the `summary` function and compared to the parameter estimates from the standard regression model:

```
> summary( reg.res )

Iterations = 1:2000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
alpha     0.08685 1.5141 0.019547        0.04242
beta      1.01921 0.3827 0.004941        0.01133
sigma     1.33536 0.8064 0.010411        0.03663
deviance 17.94922 4.3351 0.055966        0.19367

2. Quantiles for each variable:

            2.5%     25%      50%     75%  97.5%
alpha    -2.9151 -0.6429  0.08433  0.7934  2.940
beta      0.2597  0.8441  1.02312  1.2069  1.770
sigma     0.5447  0.8393  1.11413  1.5487  3.525
deviance 12.9271 14.7187 16.78715 20.0112 29.016

> ci.lin( m0 )

               Estimate    StdErr          z            P       2.5%    97.5%
(Intercept) 0.06666667 0.7815329 0.08530245 9.320209e-01 -1.4651096 1.598443
x           1.02857143 0.2006791 5.12545318 2.968229e-07  0.6352476 1.421895

> summary( m0 )$sigma

[1] 0.839501
```

It is seen that the ML estimates and the posterior means / medians are in fairly good agreement whereas the estimate of σ is pretty far away from the posterior mean / median. This is partly due to the fact that the dataset have 6 observations and hence virtually no information about the residual standard deviation.

3. If we try to do the parallel analysis of a real dataset with some 500 obeservations we must make sure that there are no missing values in the x-variable.

   From the `births` dataset we will use $y =$`bweight` and $x =$ `gestwks` $- 35$. We can use almost the same code as for the small bogus dataset:

```
> data( births )
> births <- subset( births, !is.na(gestwks) )
> dim( births )

[1] 490   8

> mb <- lm( bweight ~ I(gestwks-35), data=births )
> summary( mb )

Call:
lm(formula = bweight ~ I(gestwks - 35), data = births)

Residuals:
      Min        1Q    Median        3Q       Max
-1698.403  -280.136    -3.639   287.610  1382.239

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     2404.902     38.504   62.46   <2e-16
I(gestwks - 35)  196.973      8.788   22.41   <2e-16

Residual standard error: 449.7 on 488 degrees of freedom
Multiple R-squared: 0.5073,        Adjusted R-squared: 0.5062
F-statistic: 502.4 on 1 and 488 DF,  p-value: < 2.2e-16

> bth.dat <- list( x=births$gestwks-35,
+                  y=births$bweight,
+                  I=nrow(births) )
> bth.ini <- list( list( alpha=2400, beta=200, sigma=400 ),
+                  list( alpha=2300, beta=150, sigma=450 ),
+                  list( alpha=2500, beta=250, sigma=500 ) )
> bth.par <- c("alpha","beta","sigma" )
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          y[i] ~ dnorm(mu[i],tau)
+          mu[i] <- alpha + beta*x[i]
+          }
+       alpha ~ dnorm(0, 1.0E-6)
+       beta  ~ dnorm(0, 1.0E-6)
+       sigma ~ dunif(0,10000)
+       tau <- 1/pow(sigma,2)
+       }",
+       file="bth.bug" )
> bth.res <- bugs( data = bth.dat,
+                  inits = bth.ini,
+                  param = bth.par,
+                  model = "bth.bug",
+               n.chains = 3,
+                 n.iter = 20000,
+               n.burnin = 10000,
+                 n.thin = 5,
+                program = "openbugs",
+                clearWD = TRUE )

Initializing chain 1: Initializing chain 2: Initializing chain 3:

> bth.res <- mcmc.list.bugs( bth.res )
> summary( bth.res )
```

```
Iterations = 1:2000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean     SD Naive SE Time-series SE
alpha    2401.8 39.195  0.50600        0.80773
beta      197.6  8.947  0.11550        0.19494
sigma     450.8 14.835  0.19152        0.32648
deviance 7378.2  2.581  0.03332        0.05495

2. Quantiles for each variable:

           2.5%    25%    50%    75%  97.5%
alpha    2326.4 2375.6 2401.9 2428.7 2478.4
beta      179.9  191.6  197.5  203.6  215.0
sigma     423.3  440.5  450.2  460.8  480.7
deviance 7375.3 7376.3 7377.5 7379.3 7384.6
```

```
> ci.lin( mb )

                 Estimate    StdErr        z P      2.5%     97.5%
(Intercept)     2404.9021 38.504320 62.45798 0 2329.4351 2480.3692
I(gestwks - 35)  196.9726  8.788133 22.41348 0  179.7482  214.1971
```

We now get a much better accordance between the regression estimates and the posterior means / medians and also for the confidence intervals. The latter is of course because the residual standard deviation is now much more precisely determined. The moral is of course that with more data you get more precision.

3.3 Examples of the Gibbs sampler and Metropolis Hastings algorithm

1. (a) Let $\theta = (\theta_1, \theta_2)$ be the mean vector, which we know has a multivariate normal posterior distribution with mean $\mathbf{y} = (y_1, y_2)$ and covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. If we let $U = \theta_1$ and $V = \theta_2$ then we can use result (A.1) on page 579 of **BDA**, which states that $p(U|V)$ is univariate normal with

$$\begin{aligned} \mathrm{E}(U|V) &= \mathrm{E}(U) + \mathrm{cov}(V,U)\mathrm{var}(V)^{-1}(V - \mathrm{E}(V)) \\ \mathrm{var}(U|V) &= \mathrm{var}(U) - \mathrm{cov}(V,U)\mathrm{var}(V)^{-1}\mathrm{cov}(U,V)) \end{aligned}$$

Substituting in the expectations, variances and covariances *conditional on* $\mathbf{y}$ into the right hand sides of these expressions gives the following results:

$$\begin{aligned} \mathrm{E}(\theta_1|\theta_2, y) &= \mathrm{E}(\theta_1|y) + \mathrm{cov}(\theta_2, \theta_1|y)\mathrm{var}(\theta_2|y)^{-1}(\theta_2 - \mathrm{E}(\theta_2|y)) \\ &= y_1 + \rho \times 1 \times (\theta_2 - y_2) \\ &= y_1 + \rho(\theta_2 - y_2) \\ \mathrm{var}(\theta_1|\theta_2, y) &= \mathrm{var}(\theta_1|y) - \rho \times \mathrm{var}(\theta_2|y)^{-1} \times \rho \\ &= 1 - \rho \times 1 \times \rho \\ &= 1 - \rho^2. \end{aligned}$$

The result for θ_2 follows by symmetry.

(b) Gibbs Sampler.

2. For the Metropolis-Hastings bivariate proposal distribution example, here's some summary plots of the sample paths.

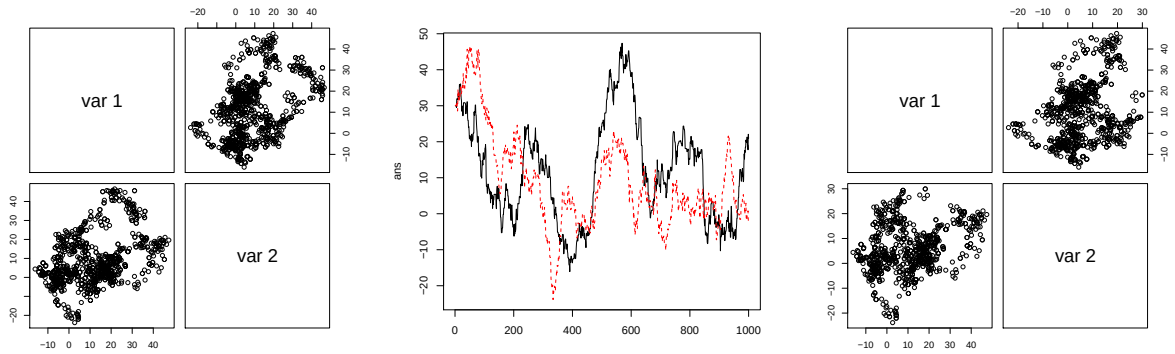

Figure 3.4: *Metropolis-Hastings sample paths*

A plot of the dependencies using the `pacf` and `acf` functions.

The acceptance probability increases slightly as the correlation parameter decreases since the proposal distribution is getting closer to the target distribution.

3. For the single component Metropolis–Hastings sampler, here's some summary plots of the sample paths.

And a plot of the acceptance probabilities:

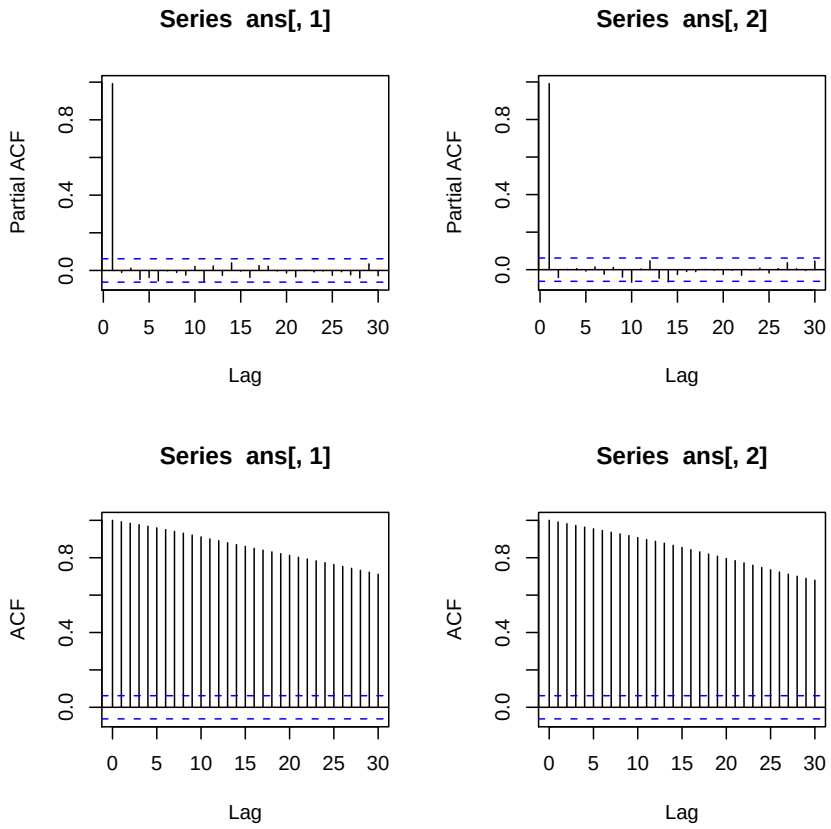

Figure 3.5: *Metropolis-Hastings — autocorrelations*

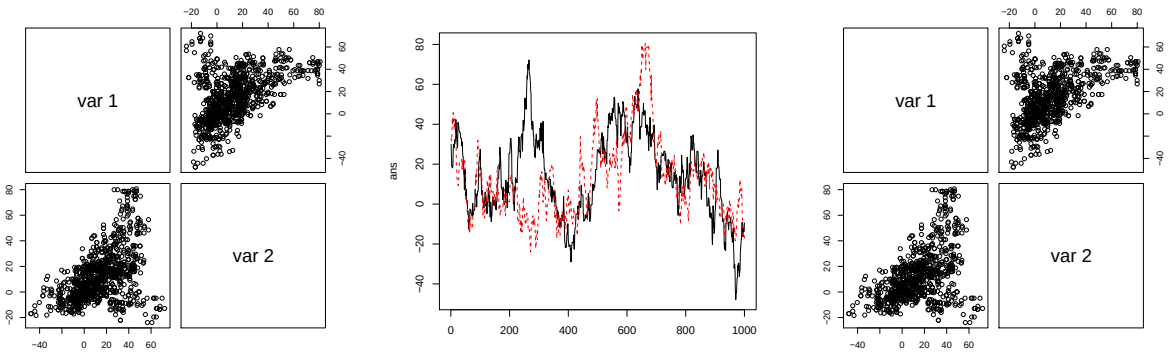

Figure 3.6: *Single component Metropolis-Hastings — sample paths*

Plotting the two series `x1` and `x2` against each other in a scatter plot is a good way to see how the length of the jumps depends on the standard deviation of the proposal distribution. The jumps get longer when the standard deviation of the proposal distribution increases.

Finally we check the dependencies within each of the `x1` and `x2` series by using the `pacf` and `acf` functions.

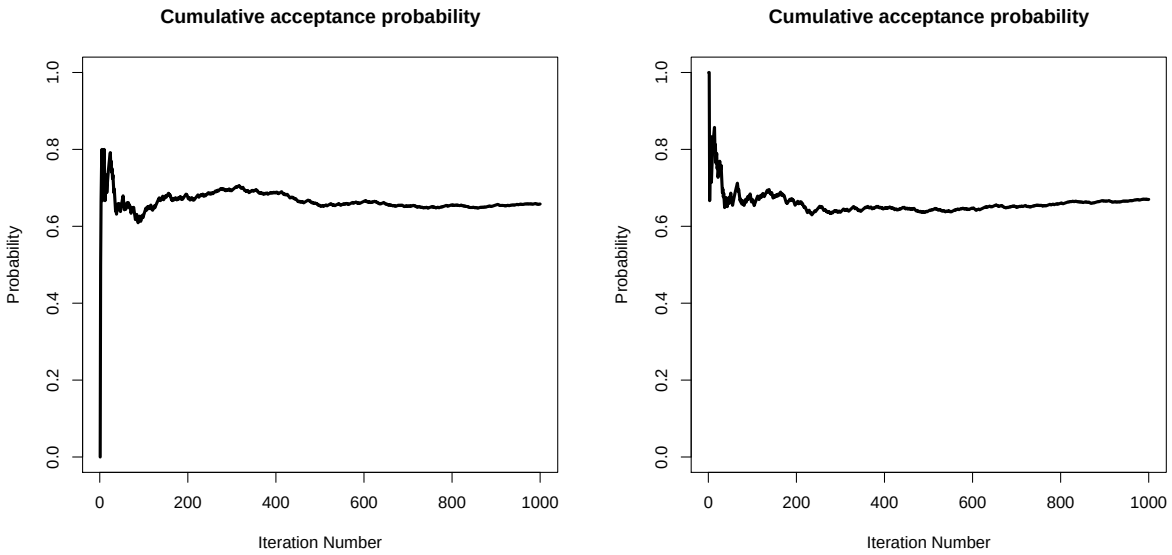

Figure 3.7: *Metropolis-Hastings acceptance probabilities*

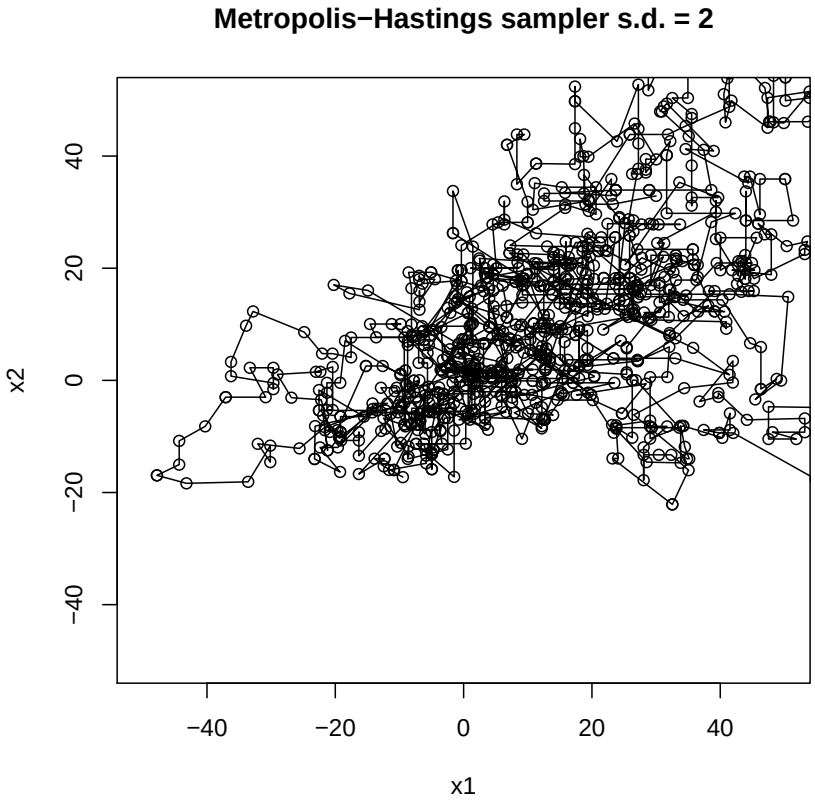

Figure 3.8: *Scatter plot of x1 and x2.*

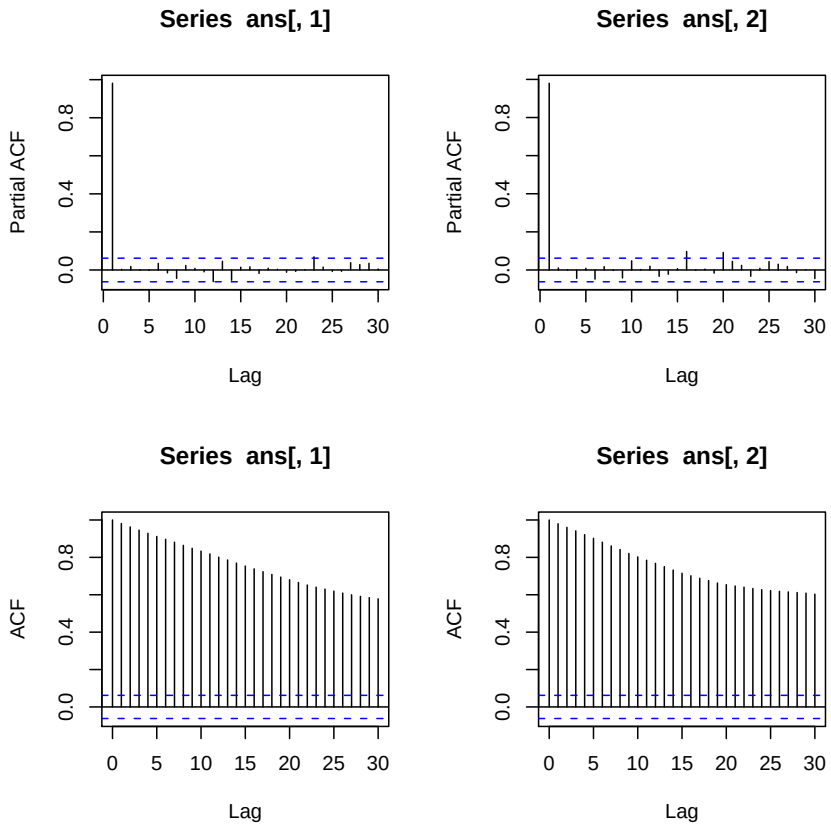

Figure 3.9: *Single component Metropolis-Hastings — autocorrelations*

3.4 Estimating a rate from Poisson data

Asthma deaths in Australia (*cf* Section 2.7 of *Bayesian Data Analysis* pages 53-55).

1. We need to work out the posterior distribution for θ based on this data. Using results from lectures (pages 53-54 of Gelman *et al.*), we find that the posterior distribution using a gamma(3,5) prior will be gamma(3+y,5+x) where $y = 3$ is the number of deaths and $x = 2$ is the number of people (in units of 100,000 since θ is expressed in these units). Thus the posterior is gamma(6,7). The posterior probability that $\theta > 2$ based on the gamma(3,5) prior and data of 3 deaths in 200,000 people is very low, about 0.5%. This was calculated using the `BUGS` code in the file `asthma.odc` and is based on 5,000 burn-in iterations and 5,000 further samples for summary:

```
node     mean   sd     2.5%    median 97.5% start sample

lambda   1.705  0.7003  0.6306 1.612  3.321 5001  5000
postprob 0.0052 0.07192 0.0    0.0    0.0   5001  5000
theta    0.8527 0.3501  0.3153 0.8058 1.66  5001  5000
```

2. The relevant `BUGS` code can be found in the file `asthma2.odc`. The node `postprob` counts the number of iterations for which the sampled value of the original rate parameter in question 1 (θ_1 or `theta[1]`) is greater than the value of the parameter corresponding to the Australian rate (θ_2 or `theta[2]`). The posterior mean of this node is an estimate of the posterior probability that the difference $\theta_1 - \theta_2 > 0$, which we see from the following output is very small, only about 0.7%:

```
node       mean   sd      2.5%   median  97.5% start sample

lambda[1]  1.72   0.7045  0.6449 1.619   3.391  5001  5000
lambda[2]  390.1  19.43   353.0  389.6   429.6  5001  5000
postprob   0.007  0.08337 0.0    0.0     0.0    5001  5000
theta[1]   0.8599 0.3522  0.3225 0.8097  1.695  5001  5000
theta[2]   1.951  0.09713 1.765  1.948   2.148  5001  5000
```

3. The sampling error associated with estimating the death rate from a population of 200,000 will be very much less than the error with which we can estimate the rate in a population of 20,000,000 - but only because the rates are fairly similar and therefore the number of events recorded in the latter population is so much larger (and this is what determines the standard error of a rate estimate). So for our purposes the latter rate can be thought of as fixed since it is the former rate that contributes most of the uncertainty. The `BUGS` code in the files `asthma2.odc` allows us to execute this example, as shown above in question 2. The same gamma(3,5) prior distribution was used for the rate in each population, resulting in posterior means for the rate per 100,000 persons per year of 0.86 and 1.95 respectively. The standard deviations of these posterior distributions are 0.352 and 0.097 respectively, reinforcing the argument about larger populations with similar death rates leading to more events and less uncertainty about the underlying rate.

4. We can certainly view the seven years worth of data as i.i.d. realisations of a Poisson random variable with mean determined by the (fixed) rate theta (per 100,000) and the population size (which we will assume is constant at 20 million). To do this in WinBUGS, simply set

up the outcome y as a vector with seven components (so `y[i]` for i in 1 to 7) and declare `y[i]` $\sim$ `dpois(lambda[i])` where the mean `lambda[i] <- (n[i]/100000)*theta[i]`.

The problem with this simple model is that the number of deaths per year due to asthma is clearly decreasing over time, and there could well be enough information in the data to generate statistical evidence for such a trend. We could model the rate as a linear function of time `alpha + beta*time`, as suggested in the airline example of exercises 2.13 and 3.12 and demonstrated in exercise 6a of the current course. By examining the posterior distribution for the slope parameter beta we can determine whether there is much evidence to support a declining rate of death due to asthma in Australia in the last few years. It is straightforward to express the rate `theta[i]` in terms of time: `theta[i] <- alpha + beta*time[i]` where time is the number of years since 1997. alpha and beta can take noninformative prior distributions (see the bioassay example section 3.7 of the textbook).

If we had further information we may even consider clustering, using information about asthma deaths by region or some other categorical exposure variable across which we might expect rates to vary. This would be an ideal scenario for the use of hierarchical modelling, where we assume variation in model parameters at each level of the hierarchy (individual, region, country etc.). We'll cover this is detail in subsequent lectures and exercises.

3.5 Estimating the speed of light

Normal distribution with unknown mean and variance (Section 3.2 of *Bayesian Data Analysis* pages 77-78).

1. The 95% posterior credible interval is obtained from the t_{65} (the degrees of freedom are $n-1 = 66-1 = 65$) marginal posterior distribution of μ as $\overline{y} \pm 1.997 s/\sqrt{66} = (23.6, 28.8)$, which follows directly from the fact that a 95% credible interval for the pivotal quantity $(\mu - \overline{y})/(s/\sqrt{n})$ is (-1.997,1.997), since the 97.5% point of the t-distribution with 65 degrees of freedom is 1.997137.

2. Based on 1000 simulated values of (μ, σ^2), Gelman et al. estimated the posterior median of μ to be 26.2 and a 95% central posterior interval for μ to be $(23.6, 28.9)$, which is quite close to the analytically calculated interval. Executing the `R` code repeatedly (say a dozen times) generates values for the posterior median of μ that are usually very close to 26.2. The lower and upper limits of the credible interval are more variable, and can differ from the quoted values above by ± 0.2 across even a small number of runs (simulating 1,000 values each time).

3. Sample output from the `BUGS` model:

```
node     mean   sd     2.5%   median 97.5% start sample

mu       26.19  1.354  23.53  26.19  28.84 10001 10000
sigma    10.96  0.9943 9.238  10.89  13.08 10001 10000
smallest 0.3858 5.681  -12.38 1.003  9.92  10001 10000
```

4. The original `BUGS` code has been amended by including the syntax

   ```
   y.pred[i] ~ dnorm(mu,tau)
   ```

   immediately under the existing statement in the "for" loop defining the distribution of the observed data `y`. Since there is no observed data for the (vector) node `y.pred`, `BUGS` simulates from the specified distribution using the current sampled values of `mu` and `tau` as required. We monitor the minimum value of the predicted vector `y.pred` by defining a node called `smallest`:

   ```
   smallest <- ranked(y.pred[],1)
   ```

   where the `ranked` function sorts the elements of its first argument (in this case `y.pred`) and returns the k^{th} smallest where k is the value of its second argument. In this case $k = 1$ so the node `smallest` does indeed contain the minimum value from our predictive sample of 66 new observations. A posterior summary of this node:

   ```
   node     mean   sd    2.5%   median 97.5% start sample

   smallest 0.3858 5.681 -12.38 1.003  9.92  10001 10000
   ```

   A minimum value of -2 (the observed second smallest value in our sample) is quite likely since it falls close to the middle of the 95% range for such minimum values. However the observed minimum of -44 is very much smaller than 95% of sampled minimum values and suggest that the normal model does not do a good job of capturing the variation that Newcomb observed.

3.6 Modelling the rate of airline fatalities 1976 to 2001

1. (a) The model for the data is:

$$y_i|\theta \sim \text{Poisson}(\theta)$$

where θ is the expected number of fatal accidents in a year.

If the prior distribution for θ is $(\Gamma(\alpha, \beta)$ then the posterior distribution is $\Gamma(\alpha + n\overline{y}, \beta + n)$, where in this case $n = 26$ and $n\overline{y} = \sum_{i=1}^{26} y_i = 634$:

```
> airline <- read.csv( "../data/airline.csv" )
> str( airline )

'data.frame':        26 obs. of  5 variables:
 $ year1975: int  1 2 3 4 5 6 7 8 9 10 ...
 $ year    : int  1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 ...
 $ fatal   : int  24 25 31 31 22 21 26 20 16 22 ...
 $ miles   : num  3.86 4.30 5.03 5.48 5.81 ...
 $ rate    : num  6.21 5.81 6.17 5.66 3.78 ...

> sum( airline$fatal )

[1] 634

> dim( airline )

[1] 26  5
```

A noninformative gamma prior distribution has $(\alpha, \beta) = (0, 0)$. This is not a proper distribution — the Γ-density is:

$$f(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} \mathrm{e}^{-\beta x}$$

so setting $(\alpha, \beta) = (0, 0)$ specifies a density proportional to $1/\theta$, which is really not possible since $\int_0^{+\infty} 1/\theta \, \mathrm{d}\theta = +\infty$. A density proportional to $1/\theta$ corresponds to a flat prior on $1/\theta$.

However, *provided* the product of the prior and the likelihood results in a proper posterior distribution for θ, (which it does in this case) we can use it.

The posterior distribution is:

$$\theta|y \sim \Gamma(634, 26)$$

and thus the posterior mean for θ is $(\alpha + n\overline{y})/(\beta + n) = 634/26 = 24.385$.

(b) Let $\tilde{y}$ be the number of fatal accidents in 2002. Given θ, the predictive distribution for $\tilde{y}$ is Poisson(θ). The derivation on pages 52 and 53 of *Bayesian Data Analysis* show that the *prior* predictive distribution for y is:

$$\begin{aligned}
p(y) &= \frac{p(y|\theta)p(\theta)}{p(\theta|y)} \\
&= \frac{\text{Poisson}(y|\theta)\text{gamma}(\theta|\alpha, \beta)}{\text{gamma}(\theta|\alpha + y, \beta + 1)} \\
&= \frac{\Gamma(\alpha + y)\beta^\alpha}{\Gamma(\alpha)y!(1 + \beta)^{\alpha+y}} \\
&= \binom{\alpha + y + 1}{y} \left(\frac{\beta}{\beta + 1}\right)^\alpha \left(\frac{1}{\beta + 1}\right)^y
\end{aligned}$$

which is the *negative binomial* density:

$$y \sim \text{Neg-bin}(\alpha, \beta)$$

For the uninformative prior (*i.e.* with $(\alpha, \beta) = 0, 0$), this is actually not a distribution, but what we actually want is the *posterior* predictive distribution for the number of fatal accidents in 2002, that is, the predictive distribution conditioning on the available data from 1976 to 2001. This has the same form as $p(y)$ presented above but we must replace α and β with the posterior quantities $\alpha^\star = \alpha + n\overline{y} = 0 + 634 = 634$ and $\beta^\star = \beta + n = 0 + 26 = 26$.

(c) The posterior distribution for θ is $\theta|y \sim \text{Gamma}(634, 26)$, and the conditional distribution of $\tilde{y}$ (the number of fatal accidents in 2002) is Poisson(θ). So to simulate values of $\tilde{y}$ all we need to do is first generate a realized value from the posterior distribution of θ and secondly sample a value from the Poisson distribution using the realized value of θ as the mean. Iterating this process will generate values of $\tilde{y}$ from the posterior predictive distribution. What we are doing here is integrating numerically, using simulation, over the posterior distribution of θ.

This can actually be accomplished in R:

```
> theta <- rgamma(1000, 634, 26 )
> y.2002 <- rpois(1000,theta)
> hist( y.2002 )
```

The default histogram is not impressive; it's actually better to explicitly plot the table of the realized values for y_{2002}:

```
> plot( table(y.2002), type="h", lwd=5, lend=2, col=gray(0.5), bty="n", ylab="" )
```

(d) The model can also be specified in `BUGS`, and run using the `bugs()` function from `R2WinBUGS`. Besides the model we need starting values and a specification of data:

```
> library(BRugs)
> library(R2WinBUGS)
> source("../r/mcmc.list.bugs.r")
> cat( "model
+       {
+       for( i in 1:I )
+         {
+         fatal[i] ~ dpois(mu)
+         }
+       mu ~ dgamma(0,0)
```

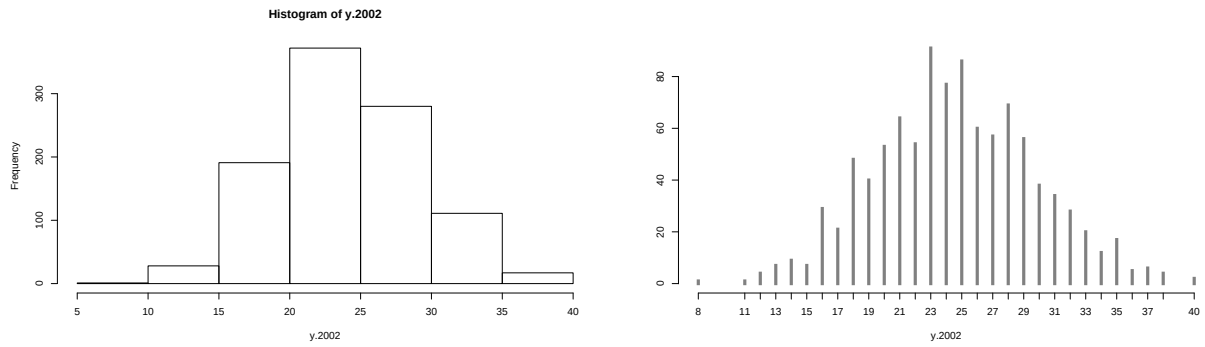

Figure 3.10: *Posterior predictive distribution of* y_{2002} *— the number of fatal airline crashes in 2002. Left panel the default* `hist()` *and right panel the result of* `plot( ..., type="h")`.

```
+       }",
+      file="a1.bug" )
> a1.ini <- list( list( mu=22 ),
+                 list( mu=23 ),
+                 list( mu=24 ) )
> a1.dat <- list( fatal = c(airline$fatal,NA), I=27 )
> a1.res <- bugs( data = a1.dat,
+                inits = a1.ini,
+                param = c("mu","fatal[27]"),
+                model = "a1.bug",
+             n.chains = 3,
+               n.iter = 30000,
+             n.burnin = 20000,
+               n.thin = 5,
+              program = "openbugs",
+                debug = FALSE,
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> # Convert the resulting bugs object, a1.res, to a mcmc.list object
> a1.mcl <- mcmc.list.bugs( a1.res )
> summary( a1.mcl )

Iterations = 1:2000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean    SD Naive SE Time-series SE
mu         24.38 0.967  0.01248        0.01411
deviance  156.23 1.374  0.01773        0.01761
fatal[27]  24.49 5.049  0.06519        0.06426

2. Quantiles for each variable:

            2.5%    25%    50%    75%  97.5%
mu         22.55  23.69  24.37  25.04  26.30
deviance  155.24 155.34 155.71 156.56 160.11
fatal[27]  15.00  21.00  24.00  28.00  35.00
```

The summary of the resulting object shows that the posterior mean and median of the μ is about 24.37. This is also the posterior expectation of the predictive distribution for the number of fatal accidents in 2002, represented by the node `fatal[27]`.

The posterior predictive distribution for the number of fatal accidents in 2002 has median 24 and 95% posterior interval [15,35]. Recall that the posterior predictive distribution is a discrete distribution. We can compare this with the one we simulated directly before:

```
> theta <- rgamma(6000, 634, 26 )
> y.2002 <- rpois(6000,theta)
> plot( table(y.2002), type="h", lwd=5, lend=2, col=gray(0.2), bty="n",
+                      ylab="", xlim=c(5,50) )
> tpr <- table( as.matrix( a1.mcl[,"fatal[27]"] ) )
> points( as.numeric(names(tpr))+0.4, tpr, type="h", col="red", lwd=4 )
```

2. (a) Let m_i = number of passenger miles flown in year i and λ = accident rate per passenger mile. The model for the data is $y_i|m_i, \lambda \sim \text{Poisson}(m_i\lambda)$. We use the noninformative $\Gamma(0, 0)$ prior distribution for λ as we did for μ previously.

The posterior distribution for λ is $\lambda|y, m \sim \Gamma(n\bar{y}, n\bar{m}) = \Gamma(634, 275.56)$ where $n\bar{m} = \sum_{i=1}^{26} m_i$:

```
> sum( airline$miles )
[1] 275.564
```

Note that the model is invariant under scaling of m in the sense that if the ms are divided by a factor K then λ is multiplied by K. In this exercise we have used the ms in the units of 10^{11}miles as they are given in the file `airline.csv`.

(b) Given λ, the predictive distribution for y_{2002} is Poisson(λm_{2002}) = Poisson$(2 \times 10^{12}\lambda)$. The posterior predictive distribution for $\tilde{y}$ will be (related to the) negative binomial but the algebra is more complex due to the presence of the 2×10^{12} scale factor based on the number of miles flown. SO we let `BUGS` do the hard work — you can see that the change to the `BUGS` code is rather minimal.

Note that we as before add an extra `NA` value to the vector of fatalities, and in order to get a predictive distribution for this an anticipated value for the number of miles flown, in this case 20 ($\times 10^{11}$).

Also note that you cannot stick an expression in as an argument to a distribution; an expression as `fatal[i] dpois(lambda*miles[i])` will cause an error.

```
> cat( "model
+       {
+       for( i in 1:I )
+         {
+         mu[i] <- lambda * miles[i]
+         fatal[i] ~ dpois( mu[i] )
+         }
+       lambda ~ dgamma(0,0)
+       }",
+      file="a2.bug" )
> a2.ini <- list( list( lambda=10 ),
+                 list( lambda=20 ),
+                 list( lambda=30 ) )
> a2.dat <- list( fatal=c(airline$fatal,NA),
+                 miles=c(airline$miles,20), I=27 )
> a2.res <- bugs( data = a2.dat,
+                inits = a2.ini,
+                param = c("lambda","fatal[27]"),
+                model = "a2.bug",
+              n.chains = 3,
+                n.iter = 30000,
```

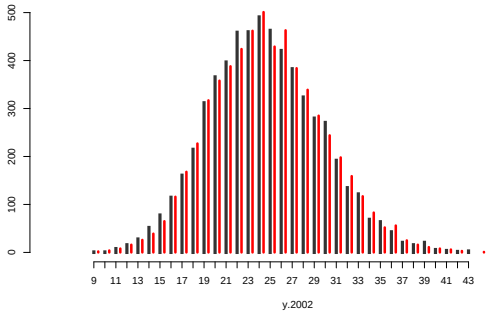

Figure 3.11: *Posterior predictive distribution of y_{2002} — the number of fatal airline crashes in 2002. Gray bars are directly simulated, red bars are the posterior from* `BUGS` *output.*

```
+             n.burnin = 20000,
+               n.thin = 5,
+              program = "openbugs",
+                debug = FALSE,
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> # Convert the resulting bugs object, a1.res, to a mcmc.list object
> a2.mcl <- mcmc.list.bugs( a2.res )
> summary( a2.mcl )

Iterations = 1:2000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean    SD Naive SE Time-series SE
lambda      2.302 0.091 0.001175       0.001197
deviance  314.089 1.421 0.018349       0.018196
fatal[27]  45.968 7.078 0.091371       0.092724

2. Quantiles for each variable:

             2.5%     25%     50%     75%   97.5%
lambda      2.131   2.238   2.301   2.363   2.481
deviance  313.100 313.191 313.566 314.439 317.985
fatal[27]  33.000  41.000  46.000  51.000  60.000
```

The posterior expectation of the predictive distribution for the number of fatal accidents in 2002 is 46 and the 95% posterior interval is [33,60].

3. (a) A closer inspection of the number of fatal airline crashes can be dome by:

```
> par(mfrow=c(1,2))
> with(airline, plot( year, fatal, pch=16, type="b", ylim=c(0,32), bty="n" ) )
> with(airline, plot( year,  rate, pch=16, type="b", ylim=c(0,7), bty="n" ) )
```

There is a decrease *on average* over the ten year period 1976 to 1985. The fatal accident *rate* per mile flown over the 26 year period shows a more consistently decreasing trend that looks amenable to modelling using a (possibly exponentially transformed) simple first order function of time.

(b) The mean of a Poisson random variable must be positive, so modelling the mean as a linear function of time, that is, $\mathrm{E}(y|\mu) = \mu = \alpha + \beta(t - 1990)$ has the potential to generate negative values for μ and thus a mean for our sampling distribution that is outside the parameter space.

In this case it seems to work, however, because the chains never get to generate a negative value of any of the `mu[i]`s:

```
> cat( "model
+       {
+       for( i in 1:I )
+         {
+         mu[i] <- (alpha + beta*(i-10)) * miles[i]
+         fatal[i] ~ dpois( mu[i] )
+         }
+       alpha ~ dnorm(0,0.000001)
+        beta ~ dnorm(0,0.000001)
+       }",
```

```
+      file="a3.bug" )
> a3.ini <- list( list( alpha=10, beta=-0.5 ),
+                 list( alpha=20, beta=-0.6 ),
+                 list( alpha=30, beta=-0.4 ) )
> a3.dat <- list( fatal=c(airline$fatal,NA),
+                 miles=c(airline$miles,20), I=27 )
> a3.res <- bugs( data = a3.dat,
+                inits = a3.ini,
+                param = c("alpha","beta","fatal[27]"),
+                model = "a3.bug",
+              n.chains = 3,
+                n.iter = 60000,
+              n.burnin = 30000,
+                n.thin = 5,
+               program = "openbugs",
+                 debug = FALSE,
+               clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> # Convert the resulting bugs object, a1.res, to a mcmc.list object
> a3.mcl <- mcmc.list.bugs( a3.res )
> summary( a3.mcl )
Iterations = 1:6000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 6000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean      SD  Naive SE Time-series SE
```

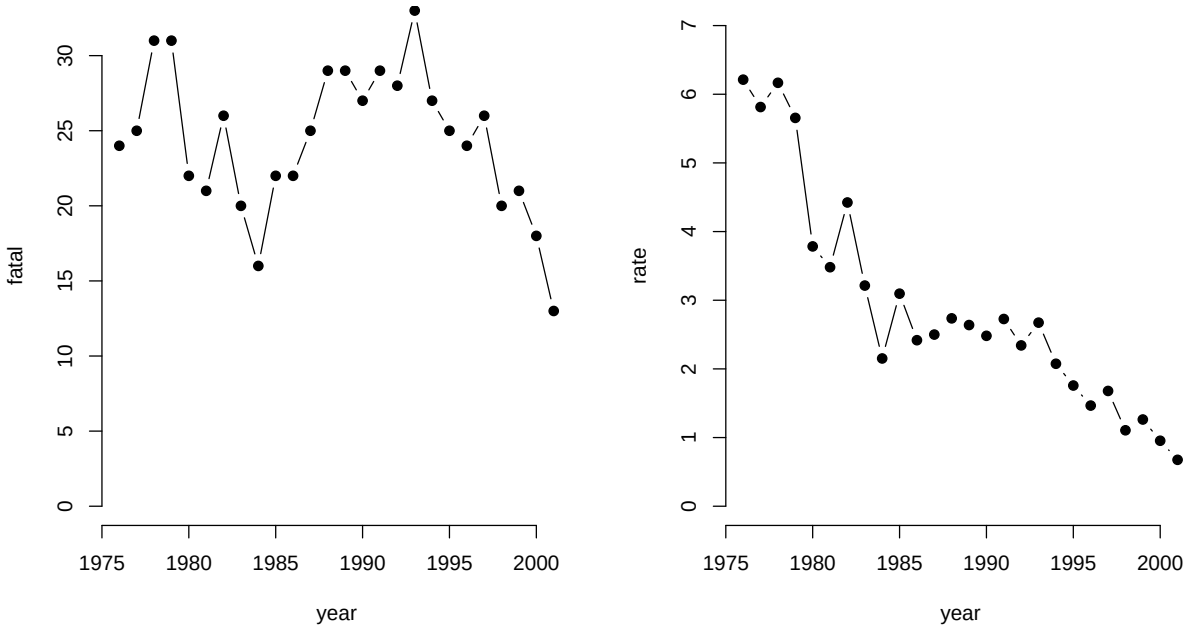

Figure 3.12: *The numbers (left) and rates (right) of fatal airline accidents.*

```
alpha       3.4442 0.15820 0.0011792      0.0103472
beta       -0.1671 0.01367 0.0001019      0.0009072
deviance  153.6645 1.99588 0.0148764      0.1094958
fatal[27]  12.1082 4.28900 0.0319683      0.1048590

2. Quantiles for each variable:

              2.5%      25%      50%      75%    97.5%
alpha       3.1423   3.3350   3.4470   3.5471   3.7767
beta       -0.1951  -0.1757  -0.1675  -0.1577  -0.1411
deviance  151.6868 152.2167 153.0656 154.4615 158.9318
fatal[27]   5.0000   9.0000  12.0000  15.0000  21.0000
```

Finally we can take a look at traces of the three chains used in this analysis (see figure 4f):

```
> print( xyplot( a3.mcl[,1:2] ) )
```

4. A more natural model is the multiplicative one

$$\log\Big(\mathrm{E}\big(y(t)|t, m(t)\big)\Big) = \alpha + \beta t + \log(m(t)) \tag{3.1}$$

   (a) The simple linear regression approach to the model is to regress the log-rate on the year:

```
> summary( lm( log( fatal/miles ) ~ I(year-1985), data=airline ) )

Call:
lm(formula = log(fatal/miles) ~ I(year - 1985), data = airline)

Residuals:
     Min       1Q   Median       3Q      Max
-0.46628 -0.14912  0.04327  0.14137  0.37938

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.163059   0.044640   26.05  < 2e-16
I(year - 1985) -0.069878   0.005394  -12.96 2.52e-12

Residual standard error: 0.2063 on 24 degrees of freedom
Multiple R-squared: 0.8749,        Adjusted R-squared: 0.8697
F-statistic: 167.8 on 1 and 24 DF,  p-value: 2.518e-12
```

   which shows that rates decrease about 7% per year ($\exp(\hat{\beta}) - 1$).

   This model puts equal weight on all observations regardless of the number of fatalities seen, so a proper Poisson-model would presumably be more appropriate.

   (b) The relevant Poisson model is one where the log of the mean is linear, as indicated in the formula (3.1) above. The log of the miles is a regression variable, but with no coefficient, *i.e.* with a regression coefficient fixed at 1. This is a so-called offset-variable:

```
> summary( glm4 <- glm( fatal ~ I(year-1985) + offset(log(miles)),
+                       family=poisson, data=airline ) )

Call:
glm(formula = fatal ~ I(year - 1985) + offset(log(miles)), family = poisson,
    data = airline)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0782  -0.7953   0.1626   0.7190   1.9370

Coefficients:
```

```
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     1.176111   0.043200   27.23   <2e-16
I(year - 1985) -0.068742   0.005394  -12.74   <2e-16

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 182.628  on 25  degrees of freedom
Residual deviance:  22.545  on 24  degrees of freedom
AIC: 157.02

Number of Fisher Scoring iterations: 4
```

This is pretty much the same results as those from the linear regression of the log-rates.

(c) We can now fit the same model using `BUGS`, by a suitable modification of the code from before:

```
> cat( "model
+       {
+       for( i in 1:I )
+         {
+         mu[i] <- exp( alpha + beta*(i-10) ) * miles[i]
+         fatal[i] ~ dpois( mu[i] )
+         }
+       alpha ~ dnorm(0,0.000001)
+        beta ~ dnorm(0,0.000001)
+       }",
+      file="a4.bug" )
> a4.ini <- list( list( alpha=1.0, beta=-0.05 ),
+                 list( alpha=1.5, beta=-0.06 ),
+                 list( alpha=0.5, beta=-0.04 ) )
> a4.dat <- list( fatal=c(airline$fatal,NA),
+                 miles=c(airline$miles,20), I=27 )
> a4.res <- bugs( data = a4.dat,
+                inits = a4.ini,
+                param = c("alpha","beta","fatal[27]"),
+                model = "a4.bug",
+              n.chains = length(a4.ini),
+                n.iter = 60000,
+              n.burnin = 30000,
+                n.thin = 5,
+               program = "openbugs",
+                 debug = FALSE,
+               clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> # Convert the resulting bugs object, a1.res, to a mcmc.list object
> a4.mcl <- mcmc.list.bugs( a4.res )
> summary( a4.mcl )
Iterations = 1:6000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 6000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

               Mean       SD  Naive SE Time-series SE
alpha       1.17481 0.043152 3.216e-04      4.004e-04
beta       -0.06884 0.005383 4.012e-05      4.519e-05
deviance  155.01529 1.968451 1.467e-02      1.714e-02
fatal[27]  20.18817 4.801404 3.579e-02      3.692e-02
```

```
2. Quantiles for each variable:

               2.5%       25%       50%       75%    97.5%
alpha       1.08897   1.14617   1.17506   1.20422   1.2588
beta       -0.07942  -0.07252  -0.06879  -0.06514  -0.0584
deviance  153.06887 153.60032 154.39702 155.80386 160.2556
fatal[27]  12.00000  17.00000  20.00000  23.00000  30.0000
```

If we compare the results with those from the generalized linear model:

```
> library( Epi )
> ci.lin( glm4 )
                   Estimate      StdErr          z P        2.5%       97.5%
(Intercept)     1.17611148 0.043199710  27.22499 0  1.09144161  1.26078136
I(year - 1985) -0.06874189 0.005393721 -12.74480 0 -0.07931339 -0.05817039
```

we see that the asymptotic 95% c.i.s from this model are virtually identical to the 95% posterior interval from the `BUGS` simulation.

(d) The mixing of the chains for α and β is checked using `xyplot` on the resulting `mcmc.list` object. This is placed alongside the corresponding plot for the model with linear trend in the rates:

```
> print( xyplot( a4.mcl[,1:2] ) )
```

(e) The mixing of the chains for α and β can also be checked by checking whether the densities based on each of the chains look similar:

```
> print( densityplot( a4.mcl[,1:2], aspect="fill" ) )
```

Likewise, we may simply plot the simulated values for α and *beta* against each other with different colors:

```
> mat4 <- as.matrix( a4.mcl, chains=TRUE )
> # permute the rows to get the colors better mixed in the plot
> mat4 <- mat4[sample(1:nrow(mat4)),]
> plot( mat4[,"alpha"], mat4[,"beta"],
+       pch=16, cex=0.3, col=rainbow(3)[mat4[,"CHAIN"]] )
```

(f) If we want the posterior of the *expected* number of airline fatalities in 2002 (assuming the the amount of flown miles is 20×10^{12}), we are asking for the posterior of $exp(\alpha + \beta \times (2002 - 1985)) \times 20$:

```
> a4.m <- as.matrix(a4.mcl)
> enum.2002 <- exp(a4.m[,"alpha"] + a4.m[,"beta"]*17)*20
> summary( enum.2002 )

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.25   18.97   20.11   20.16   21.29   27.73

> ( e2002.qnt <- quantile( enum.2002, probs=c(50,2.5,97.5)/100 ) )

     50%     2.5%    97.5%
20.10864 16.95509 23.62512
```

A plot of the posterior density of this can be obtained using the `density` function:

```
> plot( density(enum.2002), type="l", lwd=3 )
> abline( v=e2002.qnt )
```

(g) The node `fatal[27]` contains the predictive distribution for the number of fatal accidents in 2002. Its posterior mean is 20.04 (similar to that for the expected number of fatal accidents in 2002) with a standard deviation of 4.864 and 95% interval [11,30]. We can plot the distribution of this by:

```
> plot( table(a4.m["fatal[27]"]),
+       type="h", lwd=5, lend=2, col=gray(0.5), bty="n", ylab=""  )
```

As an aside, the actual figures for 2002, 2003 and 2004 are shown in table 3.1. Note that the guess that 20×10^{11} miles would be flown in 2002 was almost spot on! Secondly, the actual number of fatal accidents was 14, less than the 20 predicted from our final model in question 3, but well within the prediction interval of (11,30). Finally, the rate in 2002 (0.708) was similar to that in 2001 (0.676, which was the lowest rate for the series up to that time), but the rates in the final two year 2003 and 2004 (0.3004 and 0.4433 respectively) are about half as great as those in the previous two years. Since 1976, the rate of fatal accidents per air mile flown has decreased by an order of magnitude, that is, it is ten times lower.

(h) To produce the posterior predictive distribution of the number of fatalities in 2002, based on the maximum likelihood estimates from the generalized liner model above, we would simulate the log-rate based on an assumption of multivariate normality of the estimates, or rather based on normality of the parameter function $\alpha + \beta(2002 - 1985)$. Then we simulate a random number from this, take the exponential and multiply by 20 to get a random sample from the posterior mean. Finally we would simulate a Poisson

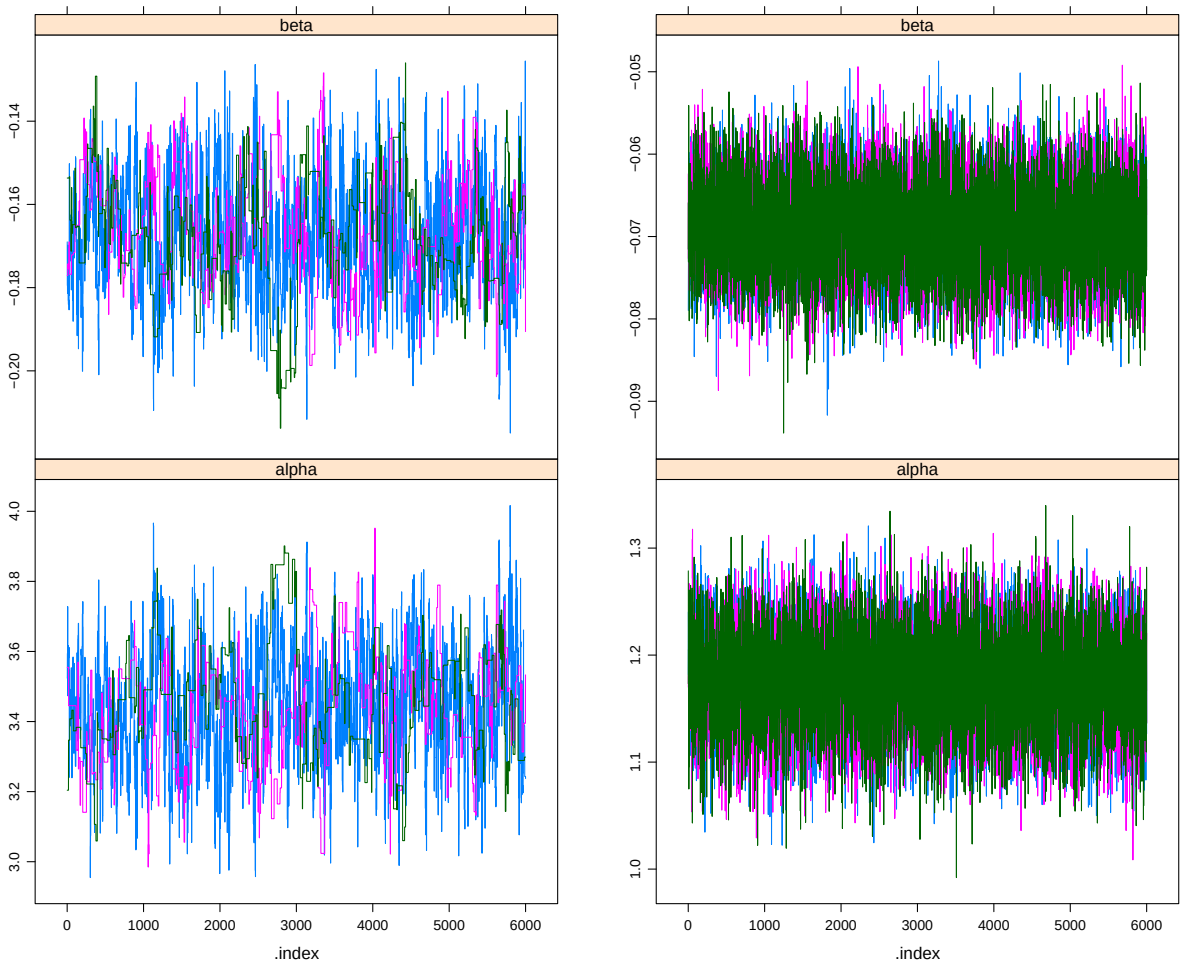

Figure 3.13: *Traceplots of chains from the linear model (left) and the log-linear model (right). For two of the chains in the linear model there is clearly some kind of boundary problems, as two of the chains stay in the same state for longer periods of time.*

Table 3.1: *Worldwide airline fatalities, 2002–2004. "Passenger miles" are in units of* 10^{11} *and the "Accident rate" is the number of fatal accidents per* 10^{11} *passenger miles. Source: International Civil Aviation Organization, Montreal, Canada (*`www.icao.int`*)*

Year	Fatal accidents	Passenger miles	Accident rate
2002	14	19.775	0.7080
2003	7	23.300	0.3004
2004	9	20.300	0.4433

variate with this mean:

```
> # ci.lin gives the estimate and its sd. for a linear combination of parameters
> mn.sd <- ci.lin( glm4, ctr.mat=rbind(c(1,2002-1985)) )[1:2]
> N <- 1000
> log.rate <- rnorm( N, mean=mn.sd[1], sd=mn.sd[2] )
> e.num <- exp( log.rate ) * 20
> p.num <- rpois( N, e.num )
> summary( p.num )
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   6.00   17.00   20.00   19.92   23.00   39.00
> quantile( p.num, probs=c(50,2.5,97.5)/100 )
  50%  2.5% 97.5%
   20    11    30
> # For comparison we make the same summary for the posterior sample
> quantile( a4.m[,"fatal[27]"], probs=c(50,2.5,97.5)/100 )
  50%  2.5% 97.5%
   20    12    30
```

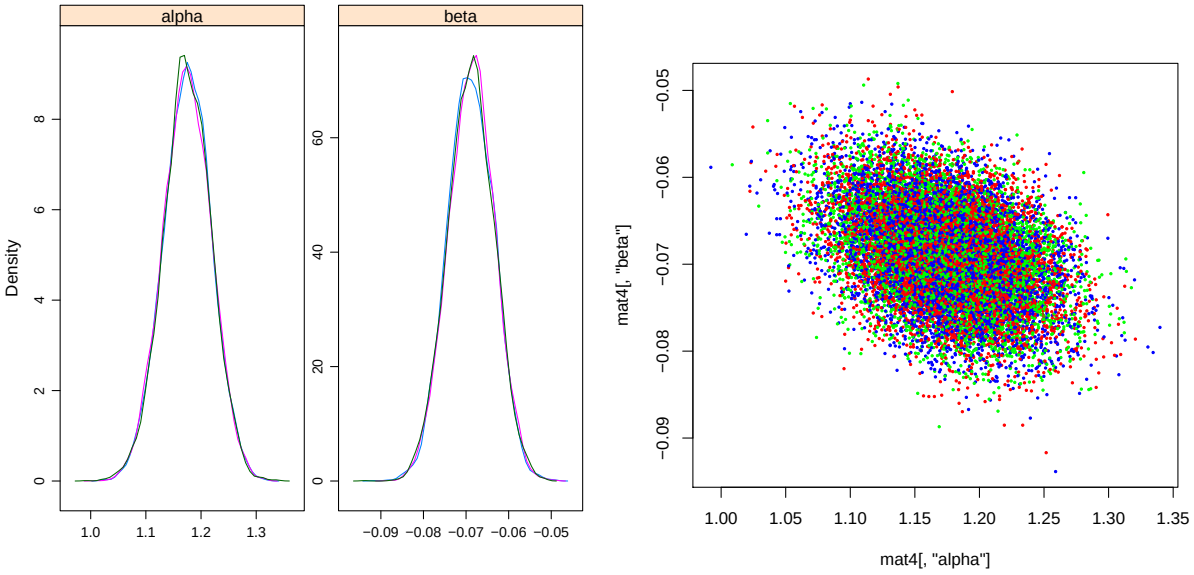

Figure 3.14: *Marginal densities (left) and joint distribution (right) for alpha and beta from the multiplicative model. Results from different chains have different colours.*

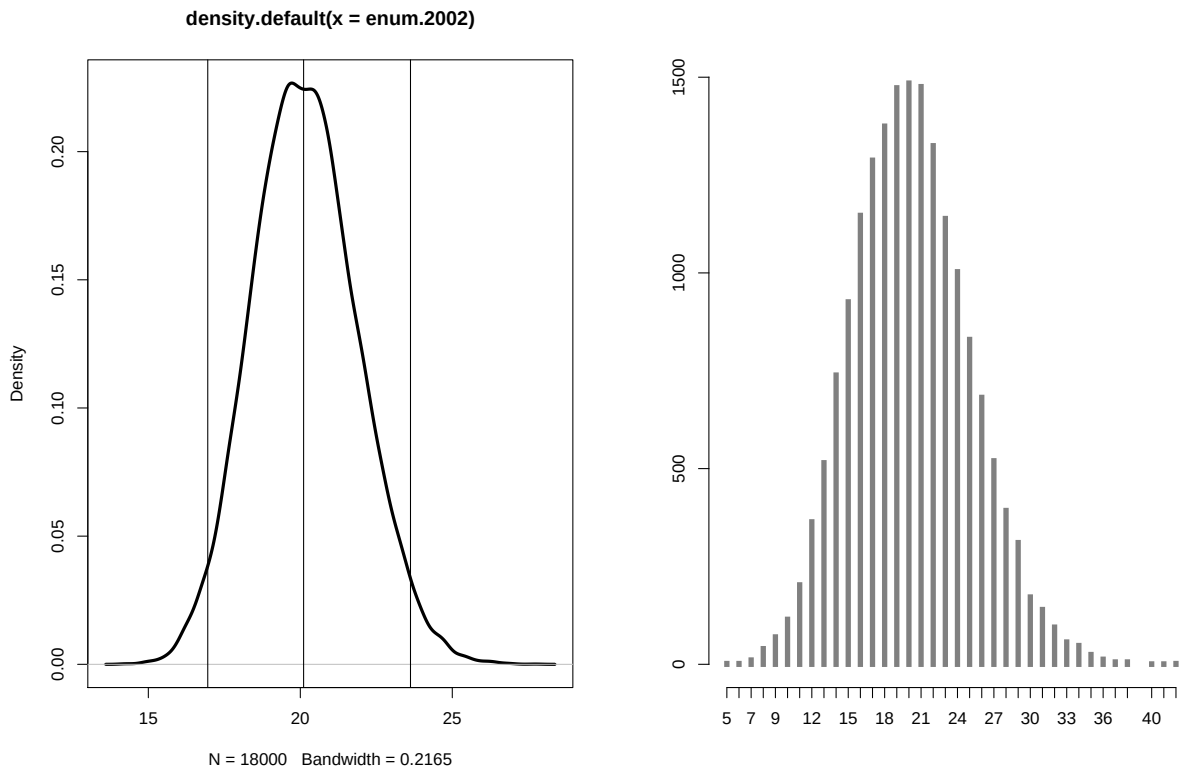

Figure 3.15: *Posterior density of the* expected *number of airline fatalities in 2002 (left) and the posterior predicted number of fatalities in 2002.*

3.7 Assessing convergence using the Gelman-Rubin diagnostic — Using `coda` in R

1. Table 3.2 shows the value of the Gelman-Rubin potential scale reduction factor $\hat{R}$ for each of 10 runs of the `schools` model on three occasions using 100, 300 and 500 iterations.

   For the simulation runs using only 100 iterations, several of the runs (*eg* numbers 2, 9 and 10) return consistently high values for $\hat{R}$ (greater than 1.2) for all nodes, and several other runs (such as run 1) have large values of $\hat{R}$ for three or more nodes. The nodes `mu.theta` and `sigma.theta` record values of $\hat{R}$ more frequently than the `theta[j]` nodes, so inference about these nodes (the parameters μ_θ and σ_θ) would benefit from a larger number of iterations.

   Increasing the number of iterations to 300 improves the values of $\hat{R}$ for the node `mu.theta` but the node `sigma.theta` still has 4 out of 10 runs with values of $\hat{R}$ that are greater than 1.2. Although this frequency decreases to only 2 out of 10 when the number of iterations is increased to 500, it is clear that the node `sigma.theta` is having the greatest trouble converging and would benefit from a larger number of iterations. In 10 runs with 1000 iterations, none had a value of $\hat{R}$ for the node `sigma.theta` greater than 1.20, but three were more than 1.15 (the values were 1.01, 1.02, 1.09, 1.10, 1.10, 1.11, 1.12, 1.15, 1.17, 1.19). This suggests that the number of iterations should be increased several times, perhaps to 5,000, to ensure proper mixing of the simulation chains.

Node	Run number									
n = 100	1	2	3	4	5	6	7	8	9	10
`theta[1]`	1.02	2.13	1.03	1.03	1.04	1.17	1.07	1.05	1.42	1.18
`theta[2]`	1.06	1.85	1.00	1.10	1.03	1.16	1.11	1.04	1.11	1.12
`theta[3]`	1.32	1.85	1.01	1.11	1.07	1.17	1.08	1.05	1.17	1.19
`theta[4]`	1.10	1.72	1.03	1.08	1.08	1.21	1.16	1.08	1.21	1.12
`theta[5]`	1.29	1.88	1.07	1.06	1.12	1.09	1.13	1.05	1.14	1.18
`theta[6]`	1.09	1.55	1.00	1.19	1.06	1.14	1.13	1.07	1.13	1.14
`theta[7]`	1.12	1.86	1.01	1.12	1.02	1.08	1.07	1.08	1.37	1.19
`theta[8]`	1.05	1.89	1.04	1.12	1.07	1.06	1.13	1.05	1.23	1.19
`mu.theta`	1.18	2.28	1.10	1.31	1.11	1.27	1.21	1.09	1.24	1.15
`sigma.theta`	1.34	1.20	1.06	1.14	1.14	1.07	1.52	1.15	3.53	1.73
n = 300										
`theta[1]`	1.03	1.03	1.03	1.04	1.10	1.01	1.06	1.03	1.04	1.02
`theta[2]`	1.03	1.04	1.02	1.03	1.10	1.04	1.03	1.01	1.06	1.02
`theta[3]`	1.07	1.02	1.04	1.04	1.15	1.06	1.04	1.02	1.05	1.03
`theta[4]`	1.06	1.05	1.01	1.08	1.13	1.04	1.10	1.01	1.01	1.04
`theta[5]`	1.08	1.02	1.04	1.04	1.30	1.04	1.03	1.03	1.02	1.02
`theta[6]`	1.07	1.02	1.03	1.04	1.16	1.04	1.06	1.01	1.01	1.04
`theta[7]`	1.06	1.06	1.04	1.06	1.10	1.03	1.06	1.06	1.01	1.03
`theta[8]`	1.04	1.02	1.02	1.06	1.10	1.03	1.05	1.05	1.03	1.03
`mu.theta`	1.10	1.04	1.02	1.08	1.11	1.10	1.09	1.07	1.06	1.06
`sigma.theta`	1.49	1.16	1.40	1.03	2.91	1.30	1.12	1.10	1.07	1.17
n = 500										
`theta[1]`	1.00	1.04	1.02	1.06	1.05	1.04	1.03	1.00	1.05	1.01
`theta[2]`	1.00	1.03	1.03	1.02	1.04	1.06	1.03	1.01	1.05	1.01
`theta[3]`	1.00	1.01	1.02	1.03	1.01	1.08	1.03	1.00	1.01	1.01
`theta[4]`	1.00	1.01	1.02	1.02	1.01	1.06	1.04	1.02	1.03	1.01
`theta[5]`	1.02	1.00	1.02	1.04	1.02	1.07	1.02	1.01	1.03	1.03
`theta[6]`	1.01	1.02	1.02	1.04	1.01	1.06	1.03	1.01	1.04	1.01
`theta[7]`	1.01	1.02	1.07	1.08	1.03	1.05	1.04	1.02	1.03	1.01
`theta[8]`	1.02	1.02	1.01	1.04	1.01	1.06	1.03	1.00	1.04	1.01
`mu.theta`	1.00	1.04	1.02	1.01	1.06	1.11	1.06	1.02	1.06	1.02
`sigma.theta`	1.00	1.09	1.08	1.33	1.12	1.29	1.01	1.04	1.06	1.15

Table 3.2: Values of the Gelman-Rubin potential scale reduction factor $\hat{R}$ from 10 runs of the `schools` model for each of 100, 300 and 500 simulations.

3.8 Meta-analysis of clinical trial data

Most of the calculations required for question 1-3 are detailed in the Microsoft Excel spreadsheet `mag_solutions.xls` although they could easily be performed in `R` as well.

1. The standard pooled-effect analysis estimates a log odds ratio of -0.4058 and a standard deviation of 0.1278, corresponding to an odds ratio of OR = 0.67 (95% credible interval from 0.52 to 0.86).

2. (a) The value of Q is 10.07056 on 8 - 1 = 7 degrees of freedom. This corresponds to a P-value of 0.185, so no evidence against the null hypothesis of homogeneity.

   (b) The value of $\hat{\tau}^2$ is 0.1258, so $\hat{\tau} = 0.3547$. Contrast this with the profile likelihood (see the question sheet and lectures for more detail and a graph), showing that the maximum likelihood estimator of τ^2 is zero.

3. It is straightforward to perform these calculations manually, and they appear in the Excel spreadsheet. The output from running the `BUGS` model is as follows

   ```
   node       mean    sd      2.5%     median   97.5%

   mu.theta -0.6182 0.266   -1.139   -0.6176 -0.09215
   theta[1] -0.6381 0.4587 -1.547   -0.6417  0.2677
   theta[2] -0.8372 0.3174 -1.458   -0.8368 -0.2078
   theta[3] -0.7569 0.423   -1.576   -0.7549  0.0608
   theta[4] -0.5774 0.4695 -1.5     -0.5773  0.3394
   theta[5] -0.2657 0.3486 -0.941   -0.2642  0.4344
   theta[6] -0.8492 0.4493 -1.734   -0.8451  0.02965
   theta[7] -0.6871 0.4522 -1.576   -0.6862  0.2027
   theta[8] -0.338   0.1406 -0.6076 -0.3394 -0.06238
   ```

   (a) The trail specific posterior means for the treatment effect have been shrunk from the empirical estimates of the log odds ratios based on the data from individual trials towards the overall effect. The extent of the shrinkage is given by the factor B_j in table 1 of the question sheet. Note also that the individual trials have narrower posterior credible intervals for the trial-specific treatment effects under the random effects model than the pooled model.

   (b) The posterior mean of μ is -0.6182 with a posterior standard deviation of 0.266. It is less precise than the estimate based on the fixed-effects model, but still is "significantly" less than 1; the estimated odds ratio is 0.54 (95% interval from 0.32 to 0.90).

4. The output from re-compiling the `BUGS` model with a uniform(0,1000) prior distribution on τ are shown below:

   ```
   node       mean    sd       2.5%     median   97.5%

   mu.theta -0.6413 0.3718 -1.483   -0.5951 -0.01174
   tau.theta 0.5976 0.4574 0.02919   0.5112   1.699
   theta[1] -0.6536 0.6152 -2.022   -0.5818   0.529
   theta[2] -0.8286 0.3751 -1.604   -0.8099 -0.1641
   theta[3] -0.8167 0.5398 -2.025   -0.7377   0.06955
   ```

```
theta[4] -0.5305 0.6321 -1.879  -0.5127  0.8285
theta[5] -0.1833 0.4061 -0.8839 -0.2314  0.7015
theta[6] -1.036  0.7195 -2.801  -0.8886 -0.02034
theta[7] -0.7489 0.6128 -2.203  -0.6561  0.3282
theta[8] -0.3378 0.1431 -0.6143 -0.3361 -0.05968
```

There has been minimal effect on the posterior mean of μ_θ, which has moved from -0.6182 to -0.6413, but its posterior standard deviation has increased markedly from 0.266 to 0.3718 to reflect the additional source variability implied by the prior distribution on τ rather than assuming τ to be fixed. Values of θ furthest from μ show the greatest change in posterior mean with the new prior distribution on τ.

The posterior mean for τ is now 0.5976. somewhat larger than the method of moments estimate of 0.35 (and the maximum likelihood estimate of zero), due to the weight that the prior density for τ assigns to large values of τ. Note, however, that the posterior standard deviation for τ is 0.4574 and posterior 95% credible interval is (0.0292,1.699), suggesting that the data are consistent with values of τ close to zero or alternatively several times larger than any of the point estimates of this parameter.

5. The output from re-compiling the BUGS model with a uniform(0,1000) prior distribution on τ and a "neutral" prior distribution on μ with mean 0 and standard deviation 0.40 are shown below:

```
node       mean   sd      2.5%    median  97.5%

mu.theta  -0.3914 0.2414 -0.8611 -0.3948  0.109
tau.theta  0.5358 0.4249  0.01989 0.4421  1.64
theta[1]  -0.4807 0.5419 -1.684  -0.4372  0.5634
theta[2]  -0.7292 0.3694 -1.534  -0.6901 -0.1266
theta[3]  -0.6439 0.4924 -1.784  -0.5664  0.1762
theta[4]  -0.3564 0.576  -1.53   -0.3692  0.9451
theta[5]  -0.1253 0.3869 -0.7713 -0.1868  0.7506
theta[6]  -0.81   0.6397 -2.433  -0.6651  0.1031
theta[7]  -0.5573 0.5594 -1.861  -0.4861  0.4743
theta[8]  -0.3182 0.1391 -0.5893 -0.3205 -0.04158
```

The posterior mean for the overall treatment effect μ is now, on the log odds scale, -0.39 with a standard deviation of 0.24, corresponding to an odds ratio of exp(-0.39) = 0.68 with 95% credible interval (0.42,1.09). Note that this credible interval includes the null value of 1, and that there is a posterior probability of about 5.25% that the overall treatment effect has an odds ratio of greater than 1 and is therefore harmful. The likelihood, "neutral" prior and posterior are shown in figure 3.16. It might seem reasonable to find odds ratios below 0.5 extremely surprising (as the prior distribution for μ implies), and hence a random effects meta-analysis and a neutral but nevertheless reasonably sceptical prior that rules out large effects renders the meta-analysis somewhat unconvincing. This finding is reinforced by the comment by Yusuf (1997) that "if one assumed that only moderate sized effects were possible, the apparent large effects observed in the meta-analysis of small trials with magnesium ... should perhaps have been tempered by this general judgment. If a result appears too good to be true, it probably is."

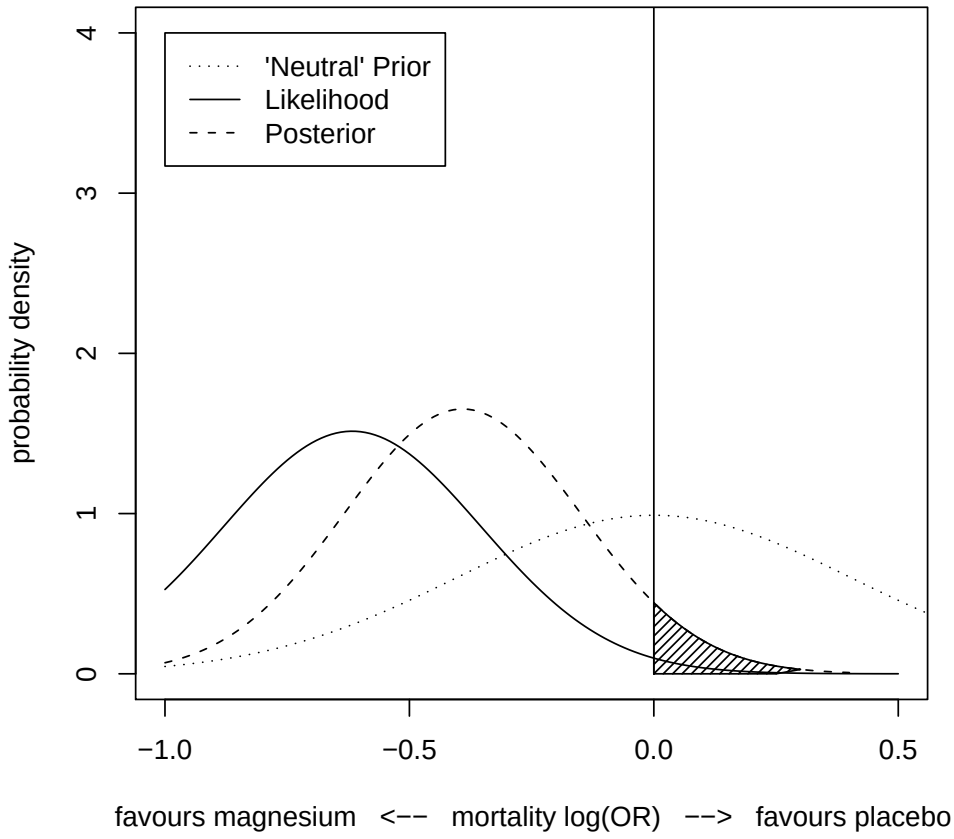

Figure 3.16: A "neutral" prior distribution for μ (mean 0 and standard deviation 0.40 on the log odd ratios scale) with the likelihood and posterior. The shaded region measures the posterior probability (about 5.25%) that treatment with magnesium is actually harmful.

3.9 Linear mixed models of fetal growth

1. A full listing of summary statistics (based on 20,000 iterations with the first 10,000 discarded as a burn-in) for all relevant nodes are as follows:

```
node              mean     sd        2.5%     median   97.5%

Sigma2.beta[1,1]  0.6596  0.0756    0.5172   0.6574   0.8148
Sigma2.beta[1,2] -0.03406 0.004041 -0.04247 -0.03388 -0.02647
Sigma2.beta[2,1] -0.03406 0.004041 -0.04247 -0.03388 -0.02647
Sigma2.beta[2,2]  0.002115 2.23E-4  0.001697 0.002106 0.00259
mu.beta[1]       -0.07925 0.04573  -0.1684  -0.07931  0.01276
mu.beta[2]        0.8681  0.002529  0.8631   0.8681   0.873
sigma.e           0.2198  0.003733  0.2125   0.2197   0.2274
sigma2.e          0.04833 0.001642  0.04517  0.04828  0.0517
```

The output from running the `lme` syntax

```
linmod <- lme(SQRTHC ~ 1 + TGA, data = hc, random = ~ 1 + TGA |
ID)
```

is as follows (using `summary(linmod)` to summarise the linear mixed model object `linmod` in `R`):

```
> summary(linmod)
Linear mixed-effects model fit by REML
 Data: hc
       AIC      BIC    logLik
  1244.966 1281.191 -616.4828

Random effects:
 Formula: ~1 + TGA | ID
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev     Corr
(Intercept) 0.81022411 (Intr)
TGA         0.04310524 -0.952
Residual    0.22146081

Fixed effects: SQRTHC ~ 1 + TGA
                 Value  Std.Error   DF  t-value p-value
(Intercept) -0.0824472 0.04439328 2390  -1.8572  0.0634
TGA          0.8683515 0.00238587 2390 363.9560  0.0000

 Correlation:
    (Intr)
TGA -0.973

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-6.19611273 -0.49096149  0.02347426  0.51250700  3.92720260
```

```
Number of Observations: 3097
Number of Groups: 706
```

Note that `lme` quotes the standard deviation instead of the variance for both the variances of the random effects and the residual error variance. The corresponding variance parameter estimates are `Sigma2.beta[1,1]` $= 0.81022411^2 = 0.6564631$, `Sigma2.beta[2,2]` $= 0.04310524^2 = 0.001858062$ and `sigma2.e` $= 0.22146081^2 = 0.04904489$. These values, along with the fixed effect estimates for the intercept `mu.beta[1]` of -0.0824472 and for the gradient `mu.beta[2]` of 0.8683515 are very similar to the posterior means of the relevant nodes displayed above in the `BUGS` summary output.

2. (a) The posterior mean for the covariance of the random effect intercept and gradient (the node `Sigma2.beta[1,2]`) is -0.03406. The posterior mean of the random effects variance for the intercept and gradient (the nodes `Sigma2.beta[1,1]` and `Sigma2.beta[2,2]` respectively) are 0.6596 and 0.002115. The estimated correlation of the random effects intercept and gradient is thus $-0.03406/\sqrt{0.6596 * 0.002115} = -0.9119$.

   The interpretation of this negative correlation with large magnitude is that fetuses with low values for the random effect (subject-specific) intercept tend to have high values for the random effect (subject-specific) gradient. That is, fetuses that start with (relatively) low head circumference (at about 18 weeks gestation) tend to show faster growth rates than those that start with (relatively) high head circumferences. An alternative interpretation is that the observed correlation structure of the data (including any heteroscedasticity) is best captured by allowing a strong negative correlation between the random effects.

   (b) Summary statistics (based on 20,000 iterations with the first 10,000 discarded as a burn-in) for the new node `rancorr` are as follows:

```
node              mean     sd         2.5%     median   97.5%

rancorr           -0.9111  0.01086  -0.9299  -0.9119  -0.8873
```

   The posterior mean for `rancorr` of -0.9111 is very close to the value of -0.9119 calculated in part (a) of the question. The 95% posterior credible interval for the correlation is (-0.9299,-0.8873), which is fairly narrow and does not suggest that there is much evidence for anything other than a negative correlation quite close to -1.

   (c) Recall that the model for Y_{ij} is

$$Y_{ij} = (\beta_0 + u_{0i}) + (\beta_1 + u_{1i})X_{ij} + \varepsilon_{ij}.$$

   If we add and subtract c from X_{ij} and re-arrange we have

$$\begin{aligned} Y_{ij} &= (\beta_0 + u_{0i}) + (\beta_1 + u_{1i})(X_{ij} - c + c) + \varepsilon_{ij} \\ &= (\beta_0 + u_{0i} + (\beta_1 + u_{1i})c) + (\beta_1 + u_{1i})(X_{ij} - c) + \varepsilon_{ij} \\ &= ((\beta_0 + \beta_1 c) + (u_{0i} + u_{1i}c)) + (\beta_1 + u_{1i})(X_{ij} - c) + \varepsilon_{ij} \\ &= (\beta_0^{'} + u_{0i}^{'}) + (\beta_1^{'} + u_{1i}^{'})X_{ij}^{'} + \varepsilon_{ij} \end{aligned}$$

   where $\beta_0^{'} = \beta_0 + \beta_1 c$, $\beta_1^{'} = \beta_1$, $u_{0i}^{'} = u_{0i} + u_{1i}c$ and $u_{1i}^{'} = u_{1i}$. The covariance between

u'_{0i} and u'_{1i} can be calculated as

$$\begin{aligned}\text{cov}(u'_{0i}, u'_{1i}) &= \text{cov}(u_{0i} + u_{1i}c, u_{1i}) \\ &= \text{cov}(u_{0i}, u_{1i}) + \text{cov}(u_{1i}c, u_{1i}) \\ &= \text{cov}(u_{0i}, u_{1i}) + c\text{var}(u_{1i}) \\ &= \sigma_{01} + c\sigma_1^2.\end{aligned}$$

So the new random effects u'_{0i} and u'_{1i} will be uncorrelated if $\sigma_{01} + c\sigma_1^2 = 0$, that is, when $c = -\sigma_{01}/\sigma_1^2$. The corresponding value of c in this case, using the posterior means of σ_{01} and σ_1^2 from above, is $c = -(-0.03406/0.002115) = 16.10$.

Note that centering the transformed gestational age (or any continuously valued covariate) at its mean removes the correlation between the estimates of the fixed effect intercept and gradient. In this case, however, the value of $c = 16.10$ required to remove the correlation between the random effects intercept and slope is not close to the mean of the transformed gestational age, which is $\overline{X} = 18.36$.

3. (a) Summary statistics for the nodes `Y[3099]` and `Y[3100]`, which contain the conditional and unconditional transformed head circumference, are as follows:

```
node      mean   sd      2.5%   median 97.5%

Y[3099]   17.59  0.4051  16.79  17.58  18.39
Y[3100]   18.33  0.4668  17.4   18.33  19.24
```

The difference in the conditional and unconditional posterior means is about 0.75, almost twice the conditional posterior standard deviation of 0.4051, and larger than the posterior standard deviation of the difference in these two means, which will be around 0.6. The fact that the conditional posterior mean (for fetus `id = 5`) is smaller than the unconditional posterior mean suggests that fetus `id = 5` has a relatively low value of transformed head circumference at 18 weeks gestation, which is reflected in the lower mean calculated for the conditional distribution of the corresponding measurement at 38 weeks.

(b) The observed value of transformed head circumference at 38.43 weeks for fetus `id = 5` is 18.38, so the conditional z-score is $z = (18.38 - 17.59)/0.4051 = 1.950136$, corresponding to the 97.44^{th} percentile of the standard normal distribution. Our interpretation of this z-score is that the transformed head circumference measurement for fetus `id = 5` was much greater than expected *given* the relatively low value of the corresponding measurement at 18 weeks. The corresponding z-score using the unconditional values for the mean and standard deviation of transformed head circumference at 38 weeks is $z = (18.38 - 18.33)/0.4668 = 0.107112$, corresponding to the 54.27^{th} percentile of the standard normal distribution. Thus the observed value for transformed head circumference for fetus `id = 5` at 38 weeks gestation is unremarkable compared to the unconditional distribution. This makes sense since a measure close to the mean of the unconditional distribution (and thus an unconditional z-score of about 0) having started from a low base should be quite unusual, as reflected by the conditional z-score of almost 2.

(c) For fetus `id = 5` whose last measurement is at gestational age 38.43 weeks, the value of the transformed gestional age is 21.20. The unconditional mean for transformed head circumference is thus $\beta_0 + \beta_1 X_{ij}$ = `mu.beta[1] + mu.beta[2]*X` = -0.07925 + 0.86871*21.20 = 18.32433, which is close to the posterior mean of 18.33 for the node

`Y[3100]` that represent the unconditional distribution of transformed head circumference at 38.43 weeks gestational age.
The variance of a single observation, is a quadratic function of transformed gestional age:

$$\begin{aligned}\text{var}(Y_{ij}) &= \text{var}(u_{0i}) + 2\text{cov}(u_{0i}, u_{1i})X_{ij} \\ &\quad +\text{var}(u_{1i})X_{ij}^2 + \text{var}(\varepsilon_{ij}) \\ &= \sigma_0^2 + 2\sigma_{01}X_{ij} + \sigma_1^2 X_{ij}^2 + \sigma_\varepsilon^2.\end{aligned}$$

The corresponding calculation with nodes from `BUGS` is

$$\text{var}(\texttt{Y}) = \texttt{Sigma2.beta}[1,1] + 2 \times \texttt{Sigma2.beta}[1,2] \times \texttt{X} + \texttt{Sigma2.beta}[2,2] \times \texttt{X}^2 + \texttt{sigma2.e}.$$

Substituting the posterior means for these nodes and the observed value of `X`, the calculated variance is
$0.6596 + 2 \times -0.03406 \times 21.20 + 0.002115 \times 21.20^2 + 0.04833 = 0.214352$. The square root of this is 0.462981 which is close to the posterior standard deviation of 0.4668 for the node `Y[3100]` that represent the unconditional distribution of transformed head circumference at 38.43 weeks gestational age.
It is possible to perform the calculations analytically (using formulae) for the conditional distribution of transformed head circumference at 38.43 weeks gestational age but they are a little more involved and we do not pursue them here.

3.10 Classical twin model in BUGS

3.10.1 Risk factors for mammographic density using twin data

1. (a) Calculations reveal that empirically $\frac{1}{2}(\text{var}(y_{i1}) + \text{var}(y_{i2})) = \frac{1}{2} \times (453.29433 + 445.32099) = 449.30766$ and that $\frac{1}{2}(\text{var}(y_{i1} - y_{i2})) = \frac{1}{2} \times 364.81535 = 182.40767$. The latter is an estimate of σ_e^2, and since theoretically $\frac{1}{2}(\text{var}(y_{i1}) + \text{var}(y_{i2})) = \sigma_a^2 + \sigma_e^2$ we can derive an estimate of σ_a^2 by subtracting $\frac{1}{2}(\text{var}(y_{i1} - y_{i2}))$ from $\frac{1}{2}(\text{var}(y_{i1}) + \text{var}(y_{i2}))$, which gives $449.30766 - 182.40767 = 266.89999$. We can generate starting values for σ_a and σ_e by taking the square root of our variance estimates, giving 16.337074 and 13.505838 respectively. The sample means of `pdens1` and `pdens2` are 37.46824 and 36.57634 respectively, so a starting value for $\mu =$ `mu` of 37 would suffice.

   (b) An output table of summary statistics appears below. The posterior mean (standard deviation) of μ is 36.99 (0.62), for σ_a^2 is 267.30 (16.31) and for σ_e^2 is 183.20 (8.63).

```
node      mean   sd      2.5%   median 97.5%

mu        36.99  0.617   35.83  36.96  38.24
sigma.a   16.34  0.4985  15.39  16.34  17.3
sigma.e   13.53  0.3186  12.95  13.53  14.17
sigma2.a  267.3  16.31   236.9  267.1  299.4
sigma2.e  183.2  8.634   167.7  182.9  200.8
```

   (c) From our model, the within-pair correlation of y_{i1} and y_{i2} is $\rho = \sigma_a^2/(\sigma_a^2 + \sigma_e^2)$. We can generate a point estimate of this correlation be replacing σ_a^2 and σ_e^2 by their posteriors means, which gives $\hat{\rho} = 267.3/(267.3 + 183.2) = 0.593341$.

2. (a) A table of posterior summary statistics for the four parameters μ, σ_a^2, σ_e^2 and $\rho_{DZ:MZ}$ based on output from the BUGS model appears below.

```
node      mean   sd      2.5%  median 97.5%

b.int     36.93  0.6109  35.85  36.88  38.18
rho       0.5973 0.05871 0.4871 0.5995 0.7032
sigma.a   18.04  0.5141  17.05  18.05  19.03
sigma.e   11.37  0.3501  10.71  11.36  12.1
sigma2.a  325.6  18.55   290.8  325.8  362.2
sigma2.e  129.4  7.977   114.6  129    146.3
```

   (b) The posterior mean of σ_a^2 has increased from 267.3 to 325.6, and the posterior mean of σ_e^2 has decreased from 183.2 to 129.4; their sum should be constant since we constrain σ_a^2 and σ_e^2 to sum to the total variation of y_{ij}. Since σ_a^2 has increased, the original model understated the within-pair correlation for MZ pairs and overstated the corresponding quantity in DZ pairs. If we subsequently establish that there are genetic factors governing mammographic density, then σ_a^2 would represent the "additive" genetic variance which would have been understated by our original, naive analysis.

   (c) The posterior mean of 0.5973 as our "best estimate" for $\rho_{DZ:MZ}$ indicates that the within-pair correlation is lower for DZ pairs than MZ pairs, which is certainly consistent with the possible influence of genetic factors on mammographic density. The posterior 95% credible interval for $\rho_{DZ:MZ}$ is (0.4871,0.7032) so it does include (only

just!) the "null" value of 0.5 corresponding to the additive genetic model. The point estimate of $\rho_{DZ:MZ} = 0.5973$ does not, however, describe an additive model.

3. (a) The (least squares) regression of percent mammographic density on age at mammogram in twin 1 produces an estimated regression coefficient of -0.797795 (s.e. 0.0784475) percent per year of age; the corresponding estimated regression coefficient in twin 2 is -0.7064001 (s.e. 0.0784852). So a starting value of -0.75 for $\beta_{\text{age}} =$ `b.age` seems like a good choice. Note that the model compiles and runs without changing the starting value of the intercept `b.int` from 37 to a more suitable value (say 76) based on a regression model of percent mammographic density (`pdens`) on age at mammogram (`agemgram`).

   (b) A summary table of the posterior distributions for the parameters μ, σ_a^2, σ_e^2, $\rho_{DZ:MZ}$ and $\beta_{\text{age}} =$ `b.age` based on output from the `BUGS` model appears below. The posterior mean of `b.age` is -0.7618 (with standard deviation 0.06778 and 95% credible interval (-0.8936,-0.6283)), so there is strong evidence against the null hypothesis that mammographic density and age are unrelated. This is consistent with inference based on the crude regression results in part (a) of the question.

node	mean	sd	2.5%	median	97.5%
b.age	-0.7618	0.06778	-0.8936	-0.7631	-0.6283
b.int	76.11	3.517	69.37	76.27	82.93
rho	0.5098	0.05045	0.4064	0.5089	0.6051
sigma.a	16.89	0.5179	15.90	16.89	17.83
sigma.e	11.26	0.3667	10.59	11.24	11.97
sigma2.a	285.4	17.49	252.8	285.4	318.1
sigma2.e	126.9	8.279	112.0	126.4	143.3

   (c) The posterior mean of $\rho_{DZ:MZ}$ is now 0.5098 (standard deviation 0.05045 with 95% posterior credible interval of (0.4064,0.6051)), which is lower than the posterior mean of 0.5973 quoted in the previous question. The posterior mean of $\rho_{DZ:MZ}$ is now very close to the value of 0.5 which would imply an additive genetic model for percent mammographic density after adjusting for age.

4. (a) The (least squares) regression of percent mammographic density on weight adjusting for age at mammogram in twin 1 produces an estimated regression coefficient of -0.6230287 (s.e. 0.0434774) percent per kg increase in weight; the corresponding estimated regression coefficient in twin 2 is -0.6647373 (s.e. 0.0449219). So a starting value of -0.64 for $\beta_{\text{weight}} =$ `b.wgt` is reasonable. The regression coefficient for age at mammogram in twin 1 is now -0.8008824 (s.e. 0.0711597) and for twin 2 is -0.7575204 (s.e. 0.0708611) so there is no need to change the starting value for $\beta_{\text{age}} =$ `b.age`.

   (b) A summary table of the posterior distributions for the parameters μ, σ_a^2, σ_e^2, $\rho_{DZ:MZ}$, $\beta_{\text{age}} =$ `b.age` and $\beta_{\text{weight}} =$ `b.wgt` based on the `BUGS` output appears below. The posterior mean of $\beta_{\text{weight}} =$ `b.wgt` is -0.6273 with standard deviation 0.03447 and posterior 95% credible interval (-0.6946,-0.5610), so there is strong evidence for a linear relationship between mammographic density and weight adjusted for age at mammogram.

node	mean	sd	2.50%	median	97.50%

```
b.age    -0.7937 0.06114 -0.91   -0.7935 -0.6694
b.int     119.8  4.081    112.1   119.8   128.1
b.wgt    -0.6273 0.03447 -0.6946 -0.6291 -0.561
rho       0.4412 0.06868  0.3145  0.4394  0.5731
sigma.a   14.88  0.4054   14.09   14.88   15.64
sigma.e   10.67  0.2974   10.08   10.66   11.23
sigma2.a  221.5  12.08    198.4   221.3   244.5
sigma2.e  113.9  6.342    101.7   113.7   126.2
```

(c) The adjustment for age changed the posterior mean of $\rho_{DZ:MZ}$ from 0.5973 to 0.5098, and the additional adjustment for weight has decreased the posterior mean further to 0.4412 (standard deviation 0.06868 and posterior 95% credible interval (0.3145,0.5731)). Although the 95% posterior credible interval overlaps the "null" value of 0.5, the point estimate (posterior mean) is no longer consistent with the additive genetic model. It has been suggested that adjusting for weight "overcorrects" the model since there is a very strong relationship between weight and non-dense area of breast tissue (recall that mammographic density is the ratio of dense area to total area = dense area + non-dense area).

3.11 Using the DIC in model comparison

1. See the graph of the data and sample means in Figure 3.17.

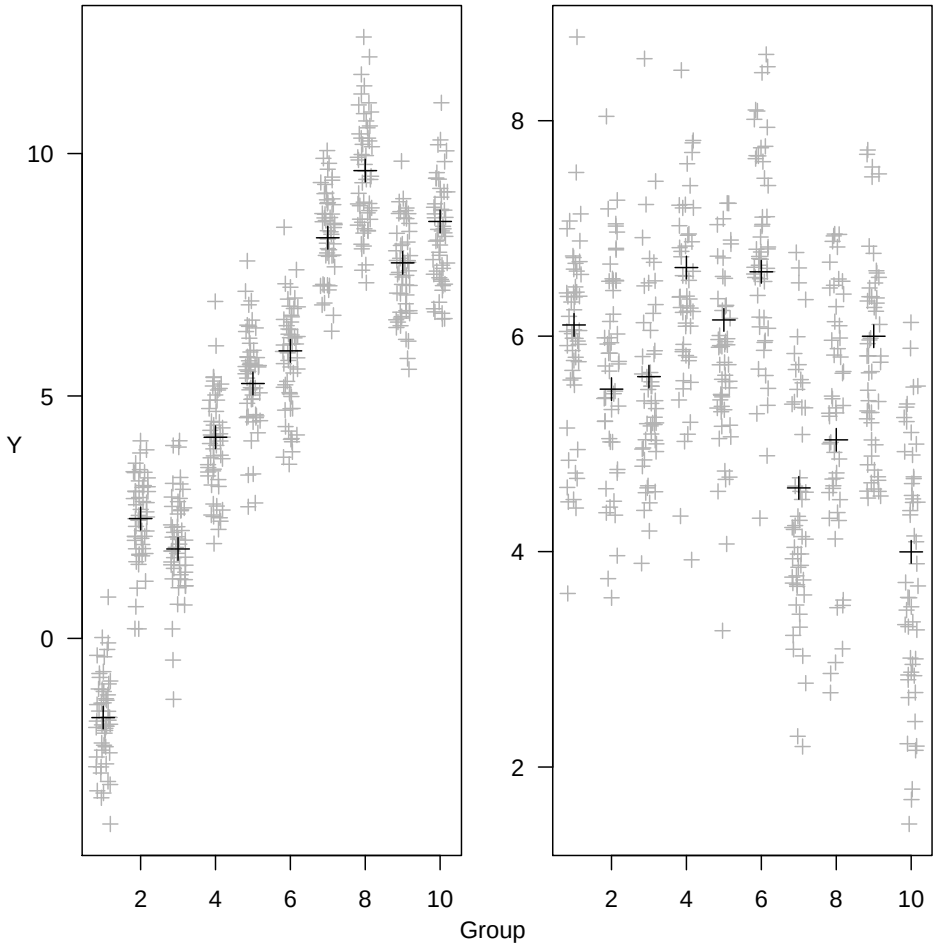

Figure 3.17: Individual data points and group-specific sample means for both datasets.

2. See Table 3.3.

Model	Data	DIC	p_D
1	1	1453.4	10.9
2	1	1453.3	10.9
1	2	1417.2	10.6
2	2	1417.6	10.9

Table 3.3: DIC and p_D for both datasets and both models.

3. See the graph of the posterior means and standard deviations of the group-specific means μ_i in Figure 3.18.

4. See Table 3.4.

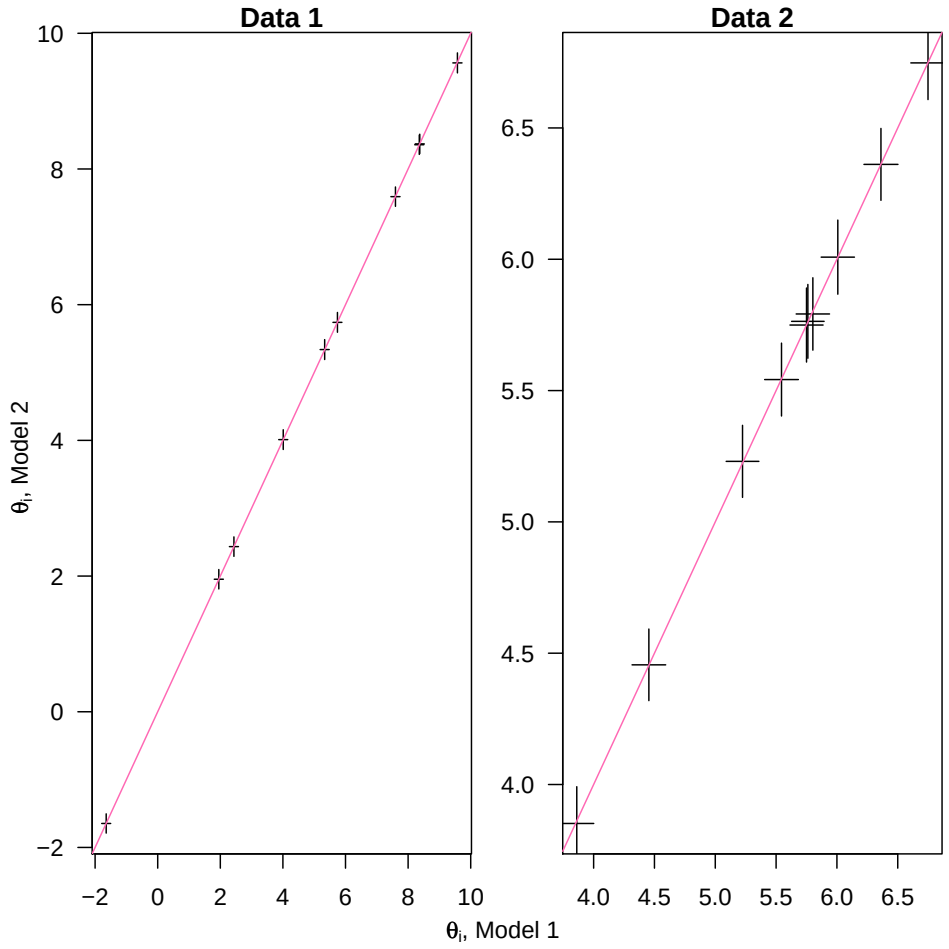

Figure 3.18: Posterior means and standard deviations of the group-specific means $\mu_i \pm$ 1 standard deviation, for each dataset and each model.

Model	Data	Post. mean τ	Post. s.d. τ
1	1	1.7	0.5
2	1	1.7	0.6
1	2	0.9	0.3
2	2	0.9	0.3

Table 3.4: Posterior mean and standard deviation of the between-group standard deviation τ for both datasets and both models.

5. See Table 3.5.

Model	Data	DIC	p_D
1	1	39.85	2.59
2	1	88.94	10.91
1	2	28.13	2.69
2	2	29.15	2.09

Table 3.5: DIC and p_D for both datasets and both models with the focus changed from θ to μ and τ.

3.12 Measurement comparison in oximetry

1. The model we consider is one where there is fixed difference between the two methods:

$$y_{(co),ir} - y_{(pulse),ir} = d_{ir} \sim \mathcal{N}(\delta, \sigma^2)$$

(a) This is just a standard normal model with mean and standard deviation as parameters, and so easily fitted in R:

```
> library( Epi )
> oxw <- read.table( "../data/ox.dat", header=TRUE )
> str(oxw)
'data.frame':        177 obs. of  4 variables:
 $ item : int  1 1 1 2 2 2 3 3 3 4 ...
 $ repl : int  1 2 3 1 2 3 1 2 3 1 ...
 $ co   : num  78 76.4 77.2 68.7 67.6 68.3 82.9 80.1 80.7 62.3 ...
 $ pulse: int  71 72 73 68 67 68 82 77 77 43 ...
> m1 <- lm( I(pulse-co) ~ 1, data=oxw )
> summary( m1 )
Call:
lm(formula = I(pulse - co) ~ 1, data = oxw)

Residuals:
     Min       1Q   Median       3Q      Max
-19.0226  -3.5226  -0.4226   3.1774  29.8774

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.4774     0.4642  -5.337 2.88e-07

Residual standard error: 6.176 on 176 degrees of freedom
```

A 95% confidence interval for the mean differnce can be found using `ci.lin` from the `Epi` package:

```
> ci.lin( m1 )
             Estimate    StdErr         z            P     2.5%     97.5%
(Intercept) -2.477401 0.4641864 -5.337083 9.445382e-08 -3.38719 -1.567613
```

(b) The prior distribution $p(\sigma^2) \propto (\sigma^2)^{-1}$ corresponds to $\nu_0 = \sigma_0^2 = 0$ so we have

$$p(\sigma^2|d) = \text{Inv-}\chi^2(n-1, s^2)$$

where $n = 177$ and s^2 is the standard deviation from the model. To obtain an observation Y from the scaled Inv-$\chi^2(n-1, s^2)$ distribution, first draw X from the χ^2_{n-1} distribution and then let $Y = (n-1)s^2/X$. The 2.5 and 97.5 percentiles of the χ^2_{n-1} distribution with $n = 177$ are found by:

```
> qchisq(c(0.025,0.975),177-1)
[1] 141.1571 214.6284
```

so a 95% posterior region for σ^2 will be the inverse of these two values multiplied by $(n-1)s^2$, so a confidence interval for σ is the square root of this:

```
> sqrt( (177-1) * summary(m1)$sigma^2 / qchisq(c(0.975,0.025),177-1) )
[1] 5.592317 6.895788
```

(c) The posterior distribution of $(\delta - \bar{d})/(s_d/\sqrt{n})$ is a t-distribution with $n-1$ degrees of freedom. So a 95% posterior interval for δ is:

$$\bar{d} \pm t_{0.975}(n-1) \times (s_d/\sqrt{n})$$

which is easily accomplished as:

```
> n <- nrow( oxw )
> coef(m1) + c(-1,1) * qt(0.975,n-1) * ( summary(m1)$sigma / sqrt(n) )
[1] -3.393489 -1.561313
```

(d) To run this in BUGS via `bugs()` we must provide, a model specification, data, initial values and the parameters to monitor:

```
> library( R2WinBUGS )
> library( BRugs )
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          d[i] ~ dnorm( delta, tausq )
+          }
+       tausq <- pow( sigma, -2 )
+       sigma ~ dunif( 0, 1000 )
+       delta ~ dnorm( 0, 0.000001 )
+       }",
+       file="m1.bug" )
> m1.dat <- list( d=oxw$co-oxw$pulse, I=nrow(oxw) )
> m1.ini <- list( list( sigma=5, delta=0 ),
+                 list( sigma=6, delta=1 ),
+                 list( sigma=4, delta=-1 ) )
> m1.par <- c("sigma","delta")
> m1.res <- bugs( data = m1.dat,
+                inits = m1.ini,
+                param = m1.par,
+                model = "m1.bug",
+            n.chains = length(m1.ini),
+              n.iter = 30000,
+            n.burnin = 20000,
+              n.thin = 10,
+             program = "openbugs",
+             clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
```

In order to summarize and check the results we need to transform the resulting `bugs` object into an `mcmc.list` object, so we must get the ad hoc function to do this:

```
> source("../r/mcmc.list.bugs.r")
> m1.res <- mcmc.list.bugs(m1.res)
```

(e) Once we have formed an `mcmc.list` object we can just use `summary` to get a 95% posterior interval for the parameters:

```
> summary( m1.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
sigma       6.222 0.3337 0.006092       0.006595
delta       2.469 0.4689 0.008561       0.010171
deviance 1147.816 2.0412 0.037268       0.042564

2. Quantiles for each variable:

             2.5%      25%      50%      75%    97.5%
```

```
sigma       5.620    5.983    6.208    6.437    6.898
delta       1.559    2.147    2.466    2.780    3.406
deviance 1145.847 1146.387 1147.224 1148.563 1153.257
```

(f) We introduce limits $\delta \pm 2\sigma$ as nodes `agree.lo` and `agree.hi` in the `BUGS` code:

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          d[i] ~ dnorm( delta, tausq )
+          }
+       tausq <- pow( sigma, -2 )
+       sigma ~ dunif( 0, 1000 )
+       delta ~ dnorm( 0, 0.000001 )
+     agree.lo <- delta - 2*sigma
+     agree.hi <- delta + 2*sigma
+       }",
+       file="m2.bug" )
> m2.dat <- list( d=oxw$co-oxw$pulse, I=nrow(oxw) )
> m2.ini <- list( list( sigma=5, delta=0 ),
+                 list( sigma=6, delta=1 ),
+                 list( sigma=4, delta=-1 ) )
> m2.par <- c("sigma","delta","agree.lo","agree.hi")
> m2.res <- bugs( data = m2.dat,
+                 inits = m2.ini,
+                 param = m2.par,
+                 model = "m2.bug",
+             n.chains = length(m2.ini),
+               n.iter = 30000,
+             n.burnin = 20000,
+               n.thin = 10,
+              program = "openbugs",
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m2.res <- mcmc.list.bugs(m2.res)
> summary( m2.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
sigma       6.222 0.3337 0.006092       0.006595
delta       2.469 0.4689 0.008561       0.010171
agree.lo   -9.975 0.8213 0.014994       0.016861
agree.hi   14.913 0.8098 0.014785       0.016268
deviance 1147.816 2.0412 0.037268       0.042564

2. Quantiles for each variable:

             2.5%      25%      50%      75%    97.5%
sigma       5.620    5.983    6.208    6.437    6.898
delta       1.559    2.147    2.466    2.780    3.406
agree.lo  -11.645  -10.509   -9.950   -9.401   -8.501
agree.hi   13.420   14.348   14.869   15.454   16.587
deviance 1145.847 1146.387 1147.224 1148.563 1153.257
```

One of the advantages of the `BUGS` machinery is that it is not necessary to re-run the

code if you want the posterior of a simple function of the parameters; we can just use the posterior sample and calculate a posterior of these parameter functions:

```
> M1 <- as.matrix( m1.res )
> a1.lo <- M1[,"delta"] - 2*M1[,"sigma"]
> a1.hi <- M1[,"delta"] + 2*M1[,"sigma"]
> M2 <- as.matrix( m2.res )
> plot( density( a1.hi ), type="l", xlim=c(-20,20), col=gray(0.5), lwd=3 )
> lines( density( a1.lo ), col=gray(0.5), lwd=3 )
> lines( density( M2[,"agree.hi"] ), lwd=2, col="red" )
> lines( density( M2[,"agree.lo"] ), lwd=2, col="red" )
```

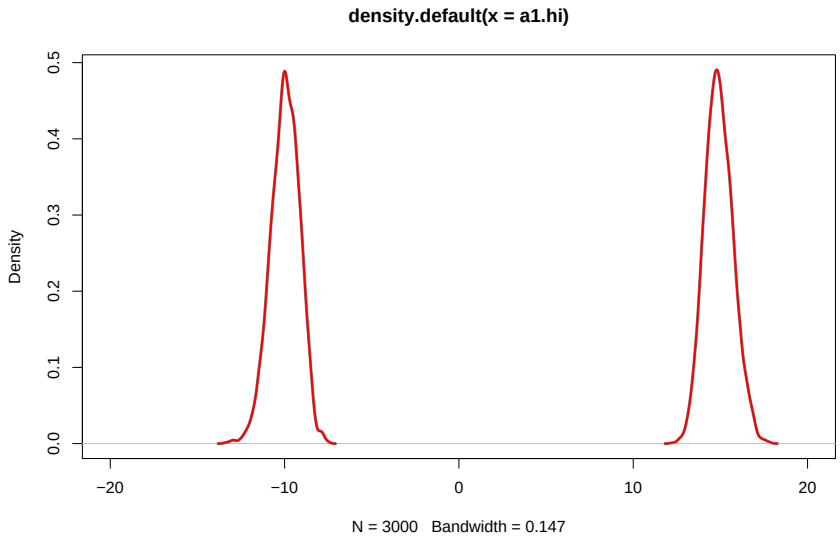

Figure 3.19: *Comparison of posterior densities for the upper and lower LoA from calculation inside* `BUGS` *(red) and from calculations on the posterior sample of* δ *and* σ

Alternatively this point could have been demonstrated using the posterior sample from model `m2` directly:

```
> summary( M2[,"agree.lo"] - (M2[,"delta"]-2*M2[,"sigma"]) )
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-9.537e-07 -2.384e-07  0.000e+00  3.179e-09  2.384e-07  9.537e-07
```

For pure numerical fun it is instructive to see a table of the deviation between the two measures:

```
> table( M2[,"agree.lo"] - (M2[,"delta"]-2*M2[,"sigma"]) )
```

```
 -9.5367431640625e-07 -8.34465026855469e-07 -7.15255737304688e-07
                   25                     7                   155
-5.96046447753906e-07 -4.76837158203125e-07 -3.57627868652344e-07
                   21                   336                    34
-2.38418579101562e-07 -1.78813934326172e-07 -1.19209289550781e-07
                  514                     1                    54
                    0  1.19209289550781e-07  1.78813934326172e-07
                  688                    46                     1
 2.38418579101562e-07  2.98023223876953e-07  3.57627868652344e-07
                  519                     1                    29
 4.76837158203125e-07  5.36441802978516e-07  5.96046447753906e-07
                  366                     1                    15
 7.15255737304688e-07  8.34465026855469e-07   9.5367431640625e-07
                  156                     9                    22
```

(g) If we instead use an informative prior corresponding to 95% in an interval 3% on either side of 0, *i.e.* $\mathcal{N}(0, 1.5^2)$, we change the BUGS code accordingly. Recall that BUGS parametrizes by the precision, i.e. the inverse variance so we use $1/1.5^2 = 0.44444$:

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          d[i] ~ dnorm( delta, tausq )
+          }
+       tausq <- pow( sigma, -2 )
+       sigma ~ dunif( 0, 1000 )
+       delta ~ dnorm( 0, 0.4444444 )
+       }",
+       file="m3.bug" )
> m3.dat <- list( d=oxw$co-oxw$pulse, I=nrow(oxw) )
> m3.ini <- list( list( sigma=5, delta=0 ),
+                 list( sigma=6, delta=1 ),
+                 list( sigma=4, delta=-1 ) )
> m3.par <- c("sigma","delta")
> m3.res <- bugs( data = m3.dat,
+                inits = m3.ini,
+                param = m3.par,
+                model = "m3.bug",
+            n.chains = length(m3.ini),
+              n.iter = 30000,
+            n.burnin = 20000,
+              n.thin = 10,
+             program = "openbugs",
+             clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m3.res <- mcmc.list.bugs(m3.res)
> summary( m3.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
sigma       6.216 0.3308 0.006040       0.005879
delta       2.261 0.4525 0.008261       0.008997
deviance 1147.950 2.1819 0.039836       0.039362

2. Quantiles for each variable:

             2.5%      25%      50%      75%    97.5%
sigma       5.623    5.986    6.201    6.430    6.897
delta       1.366    1.952    2.269    2.571    3.113
deviance 1145.857 1146.402 1147.281 1148.751 1154.189
```

We compare the posterior in this case with the previously obtained, by plotting the posterior densities on top of each other. Also we include the prior density.

```
> M3 <- as.matrix( m3.res )
> plot( density( M3[,"delta"]), type="l", col=gray(0.2), lwd=3,
+                               main="", bty="n", xlab="" )
> lines( density( M2[,"delta"] ), lwd=2, col="red" )
> xx <- seq(0,5,,200)
> lines( xx, dnorm(xx,mean=0,sd=1.5), lwd=2, col=gray(0.6) )
```

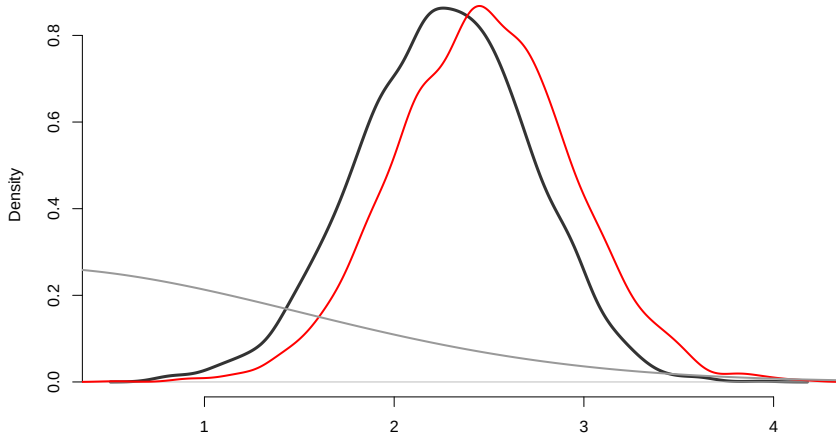

Figure 3.20: *Comparison of posterior densities using different priors for δ; informative is gray, uninformative is red. (Part of) the informative prior used is shown in light gray.*

The posterior is drawn toward 0 (the mean of the informative prior) and slightly narrower (corresponding to the larger amount of information)

2. In order to account for the individual effect of child, we introduce a *subject-specific* effect μ_i shared by all measurements on the i^{th} infant:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + \delta + e_{\text{pulse},ir} \end{aligned}$$

where $e_{mij} \sim \text{N}(0, \sigma_m^2)$, $m = \text{co}, \text{pulse}$. Note that the error terms for the two methods are different as it would rather daft to assume that the measurement error were the same for two different methods.

   (a) The distribution of $d_{ir} = y_{\text{co},ir} - y_{\text{pulse},ir}$ under this model is normal with mean δ and standard deviation $\sqrt{\sigma_{\text{co}}^2 + \sigma_{\text{co}}^2}$. So as far as the differences are concerned, the model is the same as above, but with this extended model we can actually identify the separate variances using the replicate measurements in the data.

   (b) The expansion of the model to model the two types of measurement requires a bit or rearrangement in the code. Note that the nodes `mu.co[i]` are defined as stochastic nodes, whereas `mu.pl[i]` are deterministic as a sum of two stochastic nodes.

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+         mu.co[i] ~ dnorm( 0, 0.000001 )
+         mu.pl[i] <- mu.co[i] + delta
+          y.co[i] ~ dnorm( mu.co[i], tausq.co )
+          y.pl[i] ~ dnorm( mu.pl[i], tausq.pl )
+          }
+       tausq.co <- pow( sigma.co, -2 )
+       tausq.pl <- pow( sigma.pl, -2 )
```

```
+       sigma.co ~ dunif( 0, 1000 )
+       sigma.pl ~ dunif( 0, 1000 )
+       delta ~ dnorm( 0, 0.000001 )
+       }",
+       file="m4.bug" )
> m4.dat <- list( y.co=oxw$co, y.pl=oxw$pulse, I=nrow(oxw) )
> m4.ini <- list( list( sigma.co=5, sigma.pl=5, mu.co=80, delta=0 ),
+                 list( sigma.co=6, sigma.pl=6, mu.co=70, delta=1 ),
+                 list( sigma.co=4, sigma.pl=4, mu.co=90, delta=-1 ) )
> m4.par <- c("sigma.pl","sigma.co","delta")
> m4.res <- bugs( data = m4.dat,
+                inits = m4.ini,
+                param = m4.par,
+                model = "m4.bug",
+             n.chains = length(m4.ini),
+               n.iter = 30000,
+             n.burnin = 20000,
+               n.thin = 10,
+              program = "openbugs",
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m4.res <- mcmc.list.bugs(m4.res)
> summary( m4.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean       SD Naive SE Time-series SE
sigma.pl    3.946   1.8560 0.033886        0.13814
sigma.co    4.056   1.8882 0.034474        0.14071
delta      -2.466   0.4792 0.008749        0.01304
deviance 1841.499 260.3119 4.752623       21.36103

2. Quantiles for each variable:

              2.5%      25%      50%      75%    97.5%
sigma.pl    0.5138    2.358    4.226    5.626    6.530
sigma.co    0.2767    2.579    4.545    5.683    6.519
delta      -3.4005   -2.789   -2.465   -2.141   -1.540
deviance 1146.4685 1748.780 1936.606 2018.382 2080.125
```

(c) When we get to these slightly more complicated models it is prudent to make a traceplot to ensure that the convergence i acceptable. In this case it does not really seem to be the case; it appears that the two variance components are very closely negatively correlated. Specifically the joint distribution is concentrated on a circle with radius 6, i.e. the sum of the two variances is 36, and this is pretty well determined, but the relative size of them is not.

```
> print( xyplot( m4.res[,c("delta","sigma.co","sigma.pl")],
+                aspect="fill", layout=c(3,1) ) )
> M4 <- as.matrix( m4.res, chains=TRUE )
> plot( M4[,"sigma.co"], M4[,"sigma.pl"], pch=16, col=rainbow(3)[M4[,"CHAIN"]] )
```

The simplest overview of the data can be made by the `densityplot` method which gives an overview of the monitored parameters:

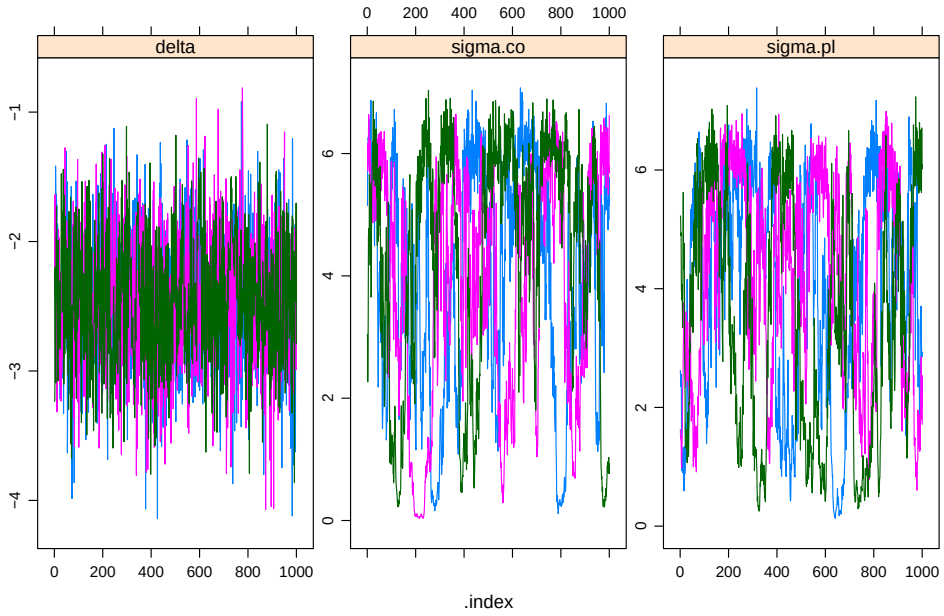

Figure 3.21: *Traces of the three chains for the three parameters of interest.*

```
> print( densityplot( m4.res[,c("delta","sigma.co","sigma.pl")],
+                     aspect="fill", layout=c(3,1) ) )
```

3. In order to account for the linking of the replicates we incorporate a random effect a_{ir} with variance ω^2, modelling the individual variation between timepoints of measurement:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + a_{ir} + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + \delta + a_{ir} + e_{\text{pulse},ir} \end{aligned}$$

   (a) We modify the BUGS code by including specification of this new variance component. In order to do this we must supply the replicate number from the data. Note the nested indexing needed in order to get the right random effect added in the right place.

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+            mu[i] ~ dunif( 0, 100 )
+         mu.co[i] <- mu[i] + a[i,repl[i]]
+         mu.pl[i] <- mu[i] + a[i,repl[i]] + delta
+          y.co[i] ~ dnorm( mu.co[i], tausq.co )
+          y.pl[i] ~ dnorm( mu.pl[i], tausq.pl )
+          for( r in 1:3 )
+             {
+             a[i,r] ~ dnorm( 0, iomegasq )
+             }
+          }
+       tausq.co <- pow( sigma.co, -2 )
+       tausq.pl <- pow( sigma.pl, -2 )
+       iomegasq <- pow( omega, -2 )
+       sigma.co ~ dunif( 0, 1000 )
+       sigma.pl ~ dunif( 0, 1000 )
+       omega    ~ dunif( 0, 1000 )
```

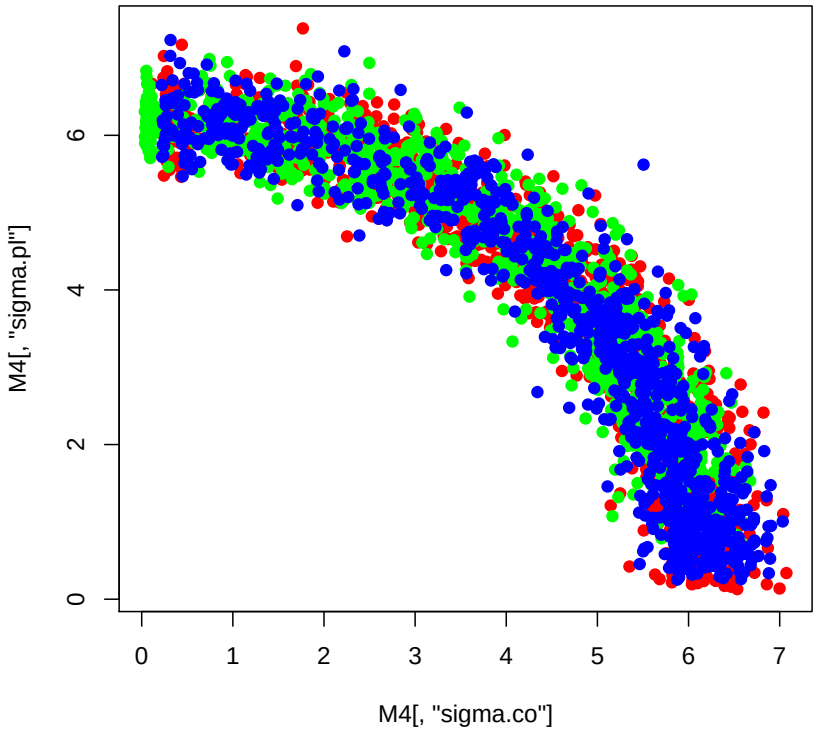

Figure 3.22: *Joint posterior distribution of the two variance components.*

```
+       delta ~ dnorm( 0, 0.000001 )
+       }",
+       file="m5.bug" )
> m5.dat <- list( y.co=oxw$co, y.pl=oxw$pulse, repl=oxw$repl, I=nrow(oxw) )
> m5.ini <- list( list( sigma.co=5, sigma.pl=5, omega=4, mu.co=80, delta=0 ),
+                 list( sigma.co=6, sigma.pl=6, omega=4, mu.co=70, delta=1 ),
+                 list( sigma.co=4, sigma.pl=4, omega=4, mu.co=90, delta=-1 ) )
> m5.par <- c("sigma.pl","sigma.co","omega","delta")
> m5.res <- bugs( data = m5.dat,
+                 inits = m5.ini,
+                 param = m5.par,
+                 model ="m5.bug",
+             n.chains = length(m5.ini),
+               n.iter = 30000,
+             n.burnin = 20000,
+               n.thin = 10,
+              program = "openbugs",
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m5.res <- mcmc.list.bugs(m5.res)
> summary( m5.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000
```

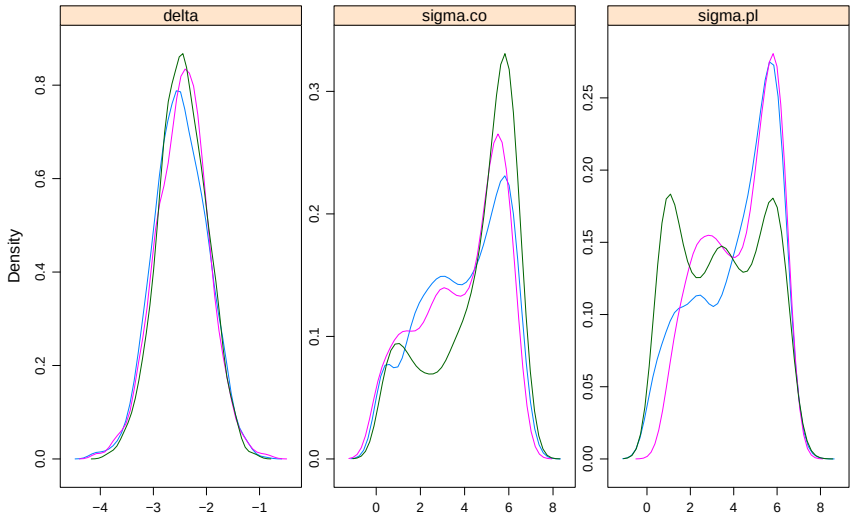

Figure 3.23: *Posterior densities for the overall difference between methods and the two residual standard deviations.*

```
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean       SD Naive SE Time-series SE
sigma.pl     3.624   1.9355 0.035337        0.14136
sigma.co     4.334   1.8175 0.033182        0.13600
omega        2.320   1.8216 0.033257        0.12913
delta       -2.470   0.4822 0.008804        0.01577
deviance 1832.773 253.0489 4.620020       19.40767

2. Quantiles for each variable:

               2.5%       25%      50%      75%    97.5%
sigma.pl    0.30950    1.8966    3.730    5.447    6.463
sigma.co    0.46824    2.9227    4.933    5.843    6.605
omega       0.04368    0.6884    2.060    3.552    6.372
delta      -3.44426   -2.7782   -2.470   -2.149   -1.531
deviance 1122.15136 1720.6963 1925.576 2014.933 2079.343
```

(b) Make a traceplot for the resulting `mcmc.list`. What is your conclusion — has the chains converged?

(c) Make a pairwise scatter plot of the parameters in the model. Use `as.matrix` to get a matrix of the posterior samples that you can stuff into `pairs`. What is your conclusion?

(d) The model can also be fitted by conventional methods, in this case we resort to `lme`. For this we first stack the data and the run the model.

```
> oxl <- data.frame( y = c(oxw$co,oxw$pulse),
+                  repl = factor( rep(oxw$repl,2) ) ,
+                    id = factor( rep(oxw$item,2) ),
+                  meth = factor( rep(c("co","pulse"),each=177) ) )
> library( nlme )
> m1 <- lme( y ~ meth + id,
+                 random = list( id = pdIdent( ~ repl-1 ) ),
+                 weights = varIdent( form = ~1 | meth ),
+                 data = oxl,
```

```
+                control = lmeControl(returnObject=TRUE) )
> m1
Linear mixed-effects model fit by REML
  Data: oxl
  Log-restricted-likelihood: -928.2544
  Fixed: y ~ meth + id
 (Intercept)    methpulse          id2          id3          id4          id5
 76.55534468  -2.47740113  -7.89502947   4.65685242 -11.28966181  -1.47555983
         id6          id7          id8          id9         id10         id11
  2.13562002   9.39463233   3.73777991  -4.99939663 -18.78304002  12.66927107
        id12         id13         id14         id15         id16         id17
-48.82331285   4.40123880  -3.66225214   6.23157059   0.48016527  13.40114335
        id18         id19         id20         id21         id22         id23
  1.48858186  -2.87219319  -1.26322060   5.64182935  -0.58513579   3.47155776
        id24         id25         id26         id27         id28         id29
  7.93409556   1.77884704   2.27263771  -9.33914552 -12.38561237   0.49639508
        id30         id31         id32         id33         id34         id35
  3.28705740 -29.97656035   5.86498335   5.75400972   8.86758775   1.12199462
        id36         id37         id38         id39         id40         id41
  3.49839611   3.56750833   6.61899307   1.73377785  -8.49118627   0.29487062
        id42         id43         id44         id45         id46         id47
 -5.97335257 -22.83052270 -17.79787217   1.82712400   4.46314117   2.91386369
        id48         id49         id50         id51         id52         id53
 -4.66545992  10.83433385 -25.14483090 -19.82772738  -0.35877402  -4.90744813
        id54         id55         id56         id57         id58         id59
 -0.05488344  11.70312835   9.29807840  12.48918523  13.11478478  14.47416217
        id60         id61
  7.63341276  -1.66927107

Random effects:
 Formula: ~repl - 1 | id
 Structure: Multiple of an Identity
          repl1   repl2   repl3 Residual
StdDev: 2.92452 2.92452 2.92452 3.005045

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | meth
 Parameter estimates:
      co    pulse
1.000000 1.795366
Number of Observations: 354
Number of Groups: 61
```

The estimates from the REML-model are $\hat{\sigma}_{\text{co}} = 3.01$ $\hat{\sigma}_{\text{pulse}} = 3.01 \times 1.795 = 5.40$ and $\omega = 2.92$, where the posterior medians are for these are 4.25, 4.47 and 2.37.

4. The simplest way to allow for a difference that varies by the true measurement levels is to introduce a linear relationship between the means:

$$\begin{aligned} y_{\text{co},ir} &= \mu_i + a_{ir} + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \alpha + \beta(\mu_i + a_{ir}) + e_{\text{pulse},ir} \end{aligned}$$

   (a) We extend the `BUGS` code by an extra mean value parameter, β, and rename the other to α, as this no longer represents a general difference between methods:

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
```

```
+            mu[i] ~ dunif( 0, 100 )
+          mu.co[i] <-                  mu[i] + a[i,repl[i]]
+          mu.pl[i] <- alpha + beta * ( mu[i] + a[i,repl[i]] )
+           y.co[i] ~ dnorm( mu.co[i], tausq.co )
+           y.pl[i] ~ dnorm( mu.pl[i], tausq.pl )
+           for( r in 1:3 )
+              {
+              a[i,r] ~ dnorm( 0, iomegasq )
+              }
+           }
+       tausq.co <- pow( sigma.co, -2 )
+       tausq.pl <- pow( sigma.pl, -2 )
+       iomegasq <- pow( omega, -2 )
+       sigma.co ~ dunif( 0, 1000 )
+       sigma.pl ~ dunif( 0, 1000 )
+       omega    ~ dunif( 0, 1000 )
+       alpha ~ dnorm( 0, 0.000001 )
+       beta  ~ dunif( 0, 2 )
+       }",
+       file="m6.bug" )
> m6.dat <- list( y.co=oxw$co, y.pl=oxw$pulse, repl=oxw$repl, I=nrow(oxw) )
> m6.ini <- list( list( sigma.co=5, sigma.pl=5, omega=4 ),
+                 list( sigma.co=6, sigma.pl=6, omega=4 ),
+                 list( sigma.co=4, sigma.pl=4, omega=4 ) )
> m6.par <- c("sigma.pl","sigma.co","omega","alpha","beta")
> m6.res <- bugs( data = m6.dat,
+                 inits = m6.ini,
+                 param = m6.par,
+                 model ="m6.bug",
+              n.chains = length(m6.ini),
+                n.iter = 30000,
+              n.burnin = 20000,
+                n.thin = 10,
+               program = "openbugs",
+               clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m6.res <- mcmc.list.bugs(m6.res)
> summary( m6.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean        SD  Naive SE Time-series SE
sigma.pl    4.2273   1.76833 0.0322851       0.122350
sigma.co    3.8568   2.03363 0.0371289       0.140788
omega       2.2648   1.57811 0.0288122       0.120445
alpha      10.7608   2.55185 0.0465901       0.155817
beta        0.8258   0.03329 0.0006078       0.002038
deviance 1836.3723 294.21028 5.3715203      21.958631

2. Quantiles for each variable:

             2.5%       25%       50%      75%     97.5%
sigma.pl   0.1194    3.3961    4.9617    5.493    6.1386
sigma.co   0.5817    2.2482    3.6062    5.606    7.3435
omega      0.1845    1.0604    1.9545    3.122    5.9657
alpha      5.7877    9.0894   10.8194   12.387   15.8701
```

```
beta       0.7603    0.8042    0.8258    0.848    0.8901
deviance 914.3640 1762.0482 1942.0375 2027.590 2097.9178
```

(b) We might as well have chosen pulse-oximetry as the reference method and re-expressed the model as

$$\begin{aligned} y_{\text{co},ir} &= \alpha^\star + \beta^\star(\mu_i + a_{ir}) + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \mu_i + a_{ir} + e_{\text{pulse},ir} \end{aligned}$$

Swapping the reference method is a pretty straightforward change to the BUGS program:

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+            mu[i] ~ dunif( 0, 100 )
+         mu.co[i] <- alpha + beta * ( mu[i] + a[i,repl[i]] )
+         mu.pl[i] <-                  mu[i] + a[i,repl[i]]
+          y.co[i] ~ dnorm( mu.co[i], tausq.co )
+          y.pl[i] ~ dnorm( mu.pl[i], tausq.pl )
+          for( r in 1:3 )
+             {
+             a[i,r] ~ dnorm( 0, iomegasq )
+             }
+          }
+       tausq.co <- pow( sigma.co, -2 )
+       tausq.pl <- pow( sigma.pl, -2 )
+       iomegasq <- pow( omega, -2 )
+       sigma.co ~ dunif( 0, 1000 )
+       sigma.pl ~ dunif( 0, 1000 )
+       omega    ~ dunif( 0, 1000 )
+       alpha ~ dnorm( 0, 0.000001 )
+       beta  ~ dunif( 0, 2 )
+       }",
+       file="m7.bug" )
> m7.dat <- list( y.co=oxw$co, y.pl=oxw$pulse, repl=oxw$repl, I=nrow(oxw) )
> m7.ini <- list( list( sigma.co=5, sigma.pl=5, omega=4 ),
+                 list( sigma.co=6, sigma.pl=6, omega=4 ),
+                 list( sigma.co=4, sigma.pl=4, omega=4 ) )
> m7.par <- c("sigma.pl","sigma.co","omega","alpha","beta")
> m7.res <- bugs( data = m7.dat,
+                inits = m7.ini,
+                param = m7.par,
+                model ="m7.bug",
+              n.chains = length(m7.ini),
+                n.iter = 30000,
+              n.burnin = 20000,
+                n.thin = 10,
+               program = "openbugs",
+               clearWD = TRUE )

Initializing chain 1: Initializing chain 2: Initializing chain 3:

> m7.res <- mcmc.list.bugs(m7.res)
> summary( m7.res )

Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

```
              Mean       SD  Naive SE Time-series SE
sigma.pl    4.4640   2.0034 0.0365776       0.141415
sigma.co    3.7948   1.9112 0.0348935       0.134226
omega       2.7086   1.9488 0.0355809       0.156487
alpha       8.2296   2.8933 0.0528234       0.204789
beta        0.9212   0.0391 0.0007139       0.002791
deviance 1846.5101 303.3006 5.5374859      22.766267

2. Quantiles for each variable:

              2.5%       25%       50%      75%     97.5%
sigma.pl   0.38172    2.8004    4.9923    6.166    7.1982
sigma.co   0.09975    2.1531    4.0669    5.554    6.4671
omega      0.18543    1.0407    2.3725    4.031    7.1426
alpha      2.53534    6.1843    8.4274   10.184   13.7898
beta       0.84469    0.8947    0.9188    0.949    0.9978
deviance 817.37809 1781.8370 1948.0836 2033.226 2106.2995
```

(c) If $\alpha + \beta\mu = \xi$ then we have $\mu = -\alpha/\beta + \xi/\beta$, hence the relationship between the parameters of the means in the two formulations are:

$$\beta^\star = 1/\beta \quad \text{and} \quad \alpha^\star = \alpha/\beta$$

(d) The `summary` function for `mcmc.list` objects allows you to extract all the relevant quantities and check whether the relationship is fulfilled for the either the mean or the median:

```
> # Mean
> ( ab6 <- summary( m6.res )$statistics[c("alpha","beta"),"Mean"] )
     alpha       beta
10.7607645  0.8258274
> ( ab7 <- summary( m7.res )$statistics[c("alpha","beta"),"Mean"] )
   alpha     beta
8.229590 0.921227
> abt <- c( -ab6[1]/ab6[2], 1/ab6[2] )
> round( cbind( ab6, ab7, abt ), 3 )
         ab6   ab7     abt
alpha 10.761 8.230 -13.030
beta   0.826 0.921   1.211
> # Median
> ( ab6 <- summary( m6.res )$quantiles[c("alpha","beta"),"50%"] )
     alpha       beta
10.8194337  0.8258169
> ( ab7 <- summary( m7.res )$quantiles[c("alpha","beta"),"50%"] )
    alpha      beta
8.4274416 0.9187737
> abt <- c( -ab6[1]/ab6[2], 1/ab6[2] )
> round( cbind( ab6, ab7, abt ), 3 )
         ab6   ab7     abt
alpha 10.819 8.427 -13.101
beta   0.826 0.919   1.211
```

Apparently the two pieces of `BUGS` code do not refer to the same model. Despite the fact that the model specifications look deceptively identical they do not give the same relationship between the models. In fact the two models are (bar the variance components) pretty close to the standard regressions of one method on the other:

```
> round(ci.lin(lm(pulse~co,data=oxw))[,c(1,5,6)],3)
            Estimate  2.5%  97.5%
(Intercept)   11.010 5.681 16.339
co             0.822 0.752  0.891
> round(summary(m6.res)$quantiles[4:5,c(3,1,5)],3)
         50%  2.5% 97.5%
alpha 10.819 5.788 15.87
beta   0.826 0.760  0.89
> round(ci.lin(lm(co~pulse,data=oxw))[,c(1,5,6)],3)
            Estimate 2.5%  97.5%
(Intercept)    8.503 2.75 14.256
pulse          0.918 0.84  0.995
> round(summary(m7.res)$quantiles[4:5,c(3,1,5)],3)
        50%  2.5%  97.5%
alpha 8.427 2.535 13.790
beta  0.919 0.845  0.998
```

5. In order to get the model right we reformulate it so that it is symmetric in the two methods:

$$\begin{aligned} y_{\text{co},ir} &= \alpha_{\text{co}} + \beta_{\text{co}}(\mu_i + a_{ir}) + e_{\text{co},ir} \\ y_{\text{pulse},ir} &= \alpha_{\text{pulse}} + \beta_{\text{pulse}}(\mu_i + a_{ir}) + e_{\text{pulse},ir} \end{aligned}$$

   (a) The relationship between the means of the two methods is found by setting all the variance components to 0 and then isolating μ_i from the first equation and inserting in the second:

$$\begin{aligned} \mu_i &= (y_{\text{co}} - \alpha_{\text{co}})/\beta_{\text{co}} \\ &\Downarrow \\ y_{\text{pulse}} &= \alpha_{\text{pulse}} + \beta_{\text{pulse}}(y_{\text{co}} - \alpha_{\text{co}})/\beta_{\text{co}} \\ &= \left(\alpha_{\text{pulse}} - \alpha_{\text{co}}\frac{\beta_{\text{pulse}}}{\beta_{\text{co}}}\right) + \frac{\beta_{\text{pulse}}}{\beta_{\text{co}}} y_{\text{co}} \end{aligned}$$

   So the relevant parameters in terms of those in the model are

$$\alpha_{\text{pulse|co}} = \alpha_{\text{pulse}} - \alpha_{\text{co}}\frac{\beta_{\text{pulse}}}{\beta_{\text{co}}} \qquad \beta_{\text{pulse|co}} = \frac{\beta_{\text{pulse}}}{\beta_{\text{co}}}$$

   (b)

   (c) The modification is quite straightforward, however it should be noted that even if the model is over-parametrized, you can still get `BUGS` to run the chains, but there is no guarantee for convergence. You might for example see the μ_is wander off to infinity and the βs going toward 0. So precisely in this case it is essential to have a finite support for the prior of the μs as this ensures a finite support for the *posterior* of the μs too.

```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+            mu[i] ~ dunif( 0, 100 )
+          mu.co[i] <- alpha.co + beta.co * ( mu[i] + a[i,repl[i]] )
+          mu.pl[i] <- alpha.pl + beta.pl * ( mu[i] + a[i,repl[i]] )
+           y.co[i] ~ dnorm( mu.co[i], tausq.co )
+           y.pl[i] ~ dnorm( mu.pl[i], tausq.pl )
```

```
+          for( r in 1:3 )
+             {
+             a[i,r] ~ dnorm( 0, iomegasq )
+             }
+          }
+       tausq.co <- pow( sigma.co, -2 )
+       tausq.pl <- pow( sigma.pl, -2 )
+       iomegasq <- pow( omega, -2 )
+       sigma.co ~ dunif( 0, 1000 )
+       sigma.pl ~ dunif( 0, 1000 )
+       omega    ~ dunif( 0, 1000 )
+       alpha.co ~ dnorm( 0, 0.000001 )
+       alpha.pl ~ dnorm( 0, 0.000001 )
+       beta.co  ~ dunif( 0, 2 )
+       beta.pl  ~ dunif( 0, 2 )
+       }",
+       file="m8.bug" )
> m8.dat <- list( y.co=oxw$co, y.pl=oxw$pulse, repl=oxw$repl, I=nrow(oxw) )
> m8.ini <- list( list( sigma.co=5, sigma.pl=5, omega=4 ),
+                 list( sigma.co=6, sigma.pl=6, omega=4 ),
+                 list( sigma.co=4, sigma.pl=4, omega=4 ) )
> m8.par <- c("sigma.pl","sigma.co","omega",
+             "alpha.pl","alpha.co",
+              "beta.pl", "beta.co")
> m8.res <- bugs( data = m8.dat,
+                inits = m8.ini,
+                param = m8.par,
+                model ="m8.bug",
+             n.chains = length(m8.ini),
+               n.iter = 30000,
+             n.burnin = 20000,
+               n.thin = 10,
+              program = "openbugs",
+              clearWD = TRUE )
Initializing chain 1: Initializing chain 2: Initializing chain 3:
> m8.res <- mcmc.list.bugs(m8.res)
> summary( m8.res )
Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean        SD  Naive SE Time-series SE
sigma.pl    3.5941   1.94711 0.0355493         0.1455
sigma.co    4.1650   1.92657 0.0351742         0.1461
omega     124.5145 101.12807 1.8463375             NA
alpha.pl   67.2601   2.86772 0.0523572             NA
alpha.co   69.4529   3.05049 0.0556941             NA
beta.pl     0.1185   0.05292 0.0009662             NA
beta.co     0.1241   0.05695 0.0010397             NA
deviance 1748.3583 383.18285 6.9959297        27.7903

2. Quantiles for each variable:

              2.5%       25%       50%       75%     97.5%
sigma.pl   0.11933 1.921e+00    4.0402    5.3225    6.1197
sigma.co   0.16206 2.706e+00    4.6947    5.8099    6.5665
omega     40.85126 6.657e+01   86.0669  136.2487  428.9874
```

```
alpha.pl  61.20678 6.559e+01   66.9888   69.4466   72.2439
alpha.co  63.06398 6.775e+01   69.3601   71.8053   74.6479
beta.pl    0.02601 7.883e-02    0.1225    0.1538    0.2288
beta.co    0.02703 8.299e-02    0.1262    0.1623    0.2416
deviance 711.36764 1.645e+03 1906.1353 2016.1856 2078.3258
```

(d) Once we have run the chains we can inspect the traces using `xyplot`; the subsetting is to get the displays in the right order — panels are filled from bottom left going left then up.

```
> print(xyplot( m8.res[,c(7,3,6,2,5,1,4)], layout=c(2,4), aspect="fill" ))
```

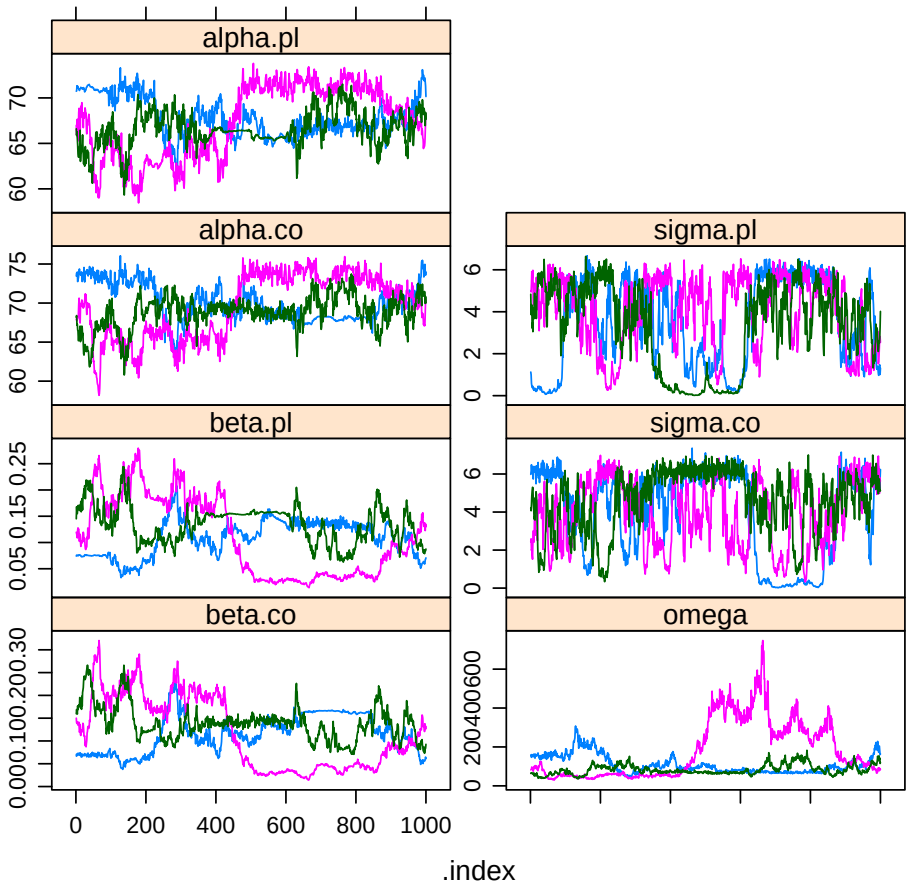

Figure 3.24: *Traces of parameters in the over-parametrized model.*

(e) The relevant parameters are the intercepts and the slopes in the linear relation between the methods. Therefore we compute these 4. Currently this is a bit of a hazzle; first convert the `mcmc.list` to a dataframe, do the camputations and turn it back into a `mcmc.list`.

```
> # Create a dataframe, expand it and coerce back to mcmc.list object:
> m8 <- as.data.frame( as.matrix( m8.res, ch=T ) )
> m8$beta.co.pl  <- m8$beta.co /  m8$beta.pl
> m8$alpha.co.pl <- m8$alpha.co - m8$alpha.pl * m8$beta.co.pl
> m8$beta.pl.co  <- m8$beta.pl /  m8$beta.co
> m8$alpha.pl.co <- m8$alpha.pl - m8$alpha.co * m8$beta.pl.co
```

```
> m8x.res <- lapply( split( as.data.frame(m8[,-1]), m8[,"CHAIN"] ),
+                    function(obj) { zz <- as.matrix(obj)
+                                    attr(zz,"mcpar") <- attr(m8.res[[1]],"mcpar")
+                                    class(zz) <- "mcmc"
+                                    return(zz) } )
> class( m8x.res ) <- "mcmc.list"
> str( m8x.res )

List of 3
 $ 1: mcmc [1:1000, 1:12] 1.126 0.997 0.592 0.469 0.388 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "1" "2" "3" "4" ...
  .. ..$ : chr [1:12] "sigma.pl" "sigma.co" "omega" "alpha.pl" ...
  ..- attr(*, "mcpar")= num [1:3] 1 1000 1
 $ 2: mcmc [1:1000, 1:12] 5.00 4.95 5.34 5.91 5.59 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "1001" "1002" "1003" "1004" ...
  .. ..$ : chr [1:12] "sigma.pl" "sigma.co" "omega" "alpha.pl" ...
  ..- attr(*, "mcpar")= num [1:3] 1 1000 1
 $ 3: mcmc [1:1000, 1:12] 4.84 3.86 4.34 4.31 3.22 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "2001" "2002" "2003" "2004" ...
  .. ..$ : chr [1:12] "sigma.pl" "sigma.co" "omega" "alpha.pl" ...
  ..- attr(*, "mcpar")= num [1:3] 1 1000 1
 - attr(*, "class")= chr "mcmc.list"

> summary( m8x.res )

Iterations = 1:1000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                Mean        SD  Naive SE Time-series SE
sigma.pl       3.5941   1.94711 0.0355493       0.145531
sigma.co       4.1650   1.92657 0.0351742       0.146069
omega        124.5145 101.12807 1.8463375             NA
alpha.pl      67.2601   2.86772 0.0523572             NA
alpha.co      69.4529   3.05049 0.0556941             NA
beta.pl        0.1185   0.05292 0.0009662             NA
beta.co        0.1241   0.05695 0.0010397             NA
deviance    1748.3583 383.18285 6.9959297      27.790316
beta.co.pl     1.0536   0.11802 0.0021547       0.007931
alpha.co.pl   -1.4468   8.65221 0.1579671       0.578963
beta.pl.co     0.9610   0.10639 0.0019423       0.007175
alpha.pl.co    0.4763   8.06735 0.1472889       0.541771

2. Quantiles for each variable:

                2.5%        25%       50%       75%     97.5%
sigma.pl     0.11933    1.92097    4.0402    5.3225    6.1197
sigma.co     0.16206    2.70602    4.6947    5.8099    6.5665
omega       40.85126   66.57066   86.0669  136.2487  428.9874
alpha.pl    61.20678   65.58865   66.9888   69.4466   72.2439
alpha.co    63.06398   67.75114   69.3601   71.8053   74.6479
beta.pl      0.02601    0.07883    0.1225    0.1538    0.2288
beta.co      0.02703    0.08299    0.1262    0.1623    0.2416
deviance   711.36764 1644.91290 1906.1353 2016.1856 2078.3258
beta.co.pl   0.87412    0.95078    1.0391    1.1545    1.2725
alpha.co.pl -17.35446   -8.74742   -0.3999    6.0358   11.8590
```

```
beta.pl.co    0.78583    0.86620    0.9623    1.0518    1.1440
alpha.pl.co -13.54335   -6.34512    0.3855    7.6182   13.6516
```

```
> round( ci.lin( lm( co ~ pulse, data=oxw ) ), 3 )
```

```
            Estimate StdErr      z     P 2.5%  97.5%
(Intercept)    8.503  2.935  2.897 0.004 2.75 14.256
pulse          0.918  0.040 23.165 0.000 0.84  0.995
```

```
> round( ci.lin( lm( pulse ~ co, data=oxw ) ), 3 )
```

```
            Estimate StdErr      z P  2.5%  97.5%
(Intercept)   11.010  2.719  4.049 0 5.681 16.339
co             0.822  0.035 23.165 0 0.752  0.891
```

We see that the slope for converting from one method to another lies between the two regression slopes we get from ordinary linear regressions.

(f) We can check whether we have reasonable mixing of the chains for the parameters of interest by `xyplot` and `density` plot — we are not impressed!

```
> wh <- c( grep( "sigma", varnames( m8x.res ) ),
+          grep( "omega", varnames( m8x.res ) ),
+          grep( "pl.co", varnames( m8x.res ) ) )
> print(xyplot( m8x.res[,wh], layout=c(3,2), aspect="fill", lwd=2 ))
```

```
> print( densityplot(m8x.res[,wh],layout=c(3,2),lwd=2,aspect="fill") )
```

(g) Based on the posterior medians we would say that the relations ship between the methods were something like:

$$y_{\text{co}} = -0.50 + 1.04 y_{\text{pulse}}$$

which is shown in the figure

```
> with( oxw, plot( co ~ pulse, pch=16, xlim=c(20,100), ylim=c(20,100) ) )
> abline(0,1)
> abline( lm( co~pulse, data=oxw), col="red", lwd=2 )
> cf <- coef( lm( pulse ~ co, data=oxw) )
> abline( -cf[1]/cf[2], 1/cf[2], col="red", lwd=2 )
> qnt <- summary( m8x.res )$quantiles
> qnt <- qnt[grep("co.pl",rownames(qnt)),"50%"]
> abline( qnt[2], qnt[1], col="blue", lwd=2 )
```

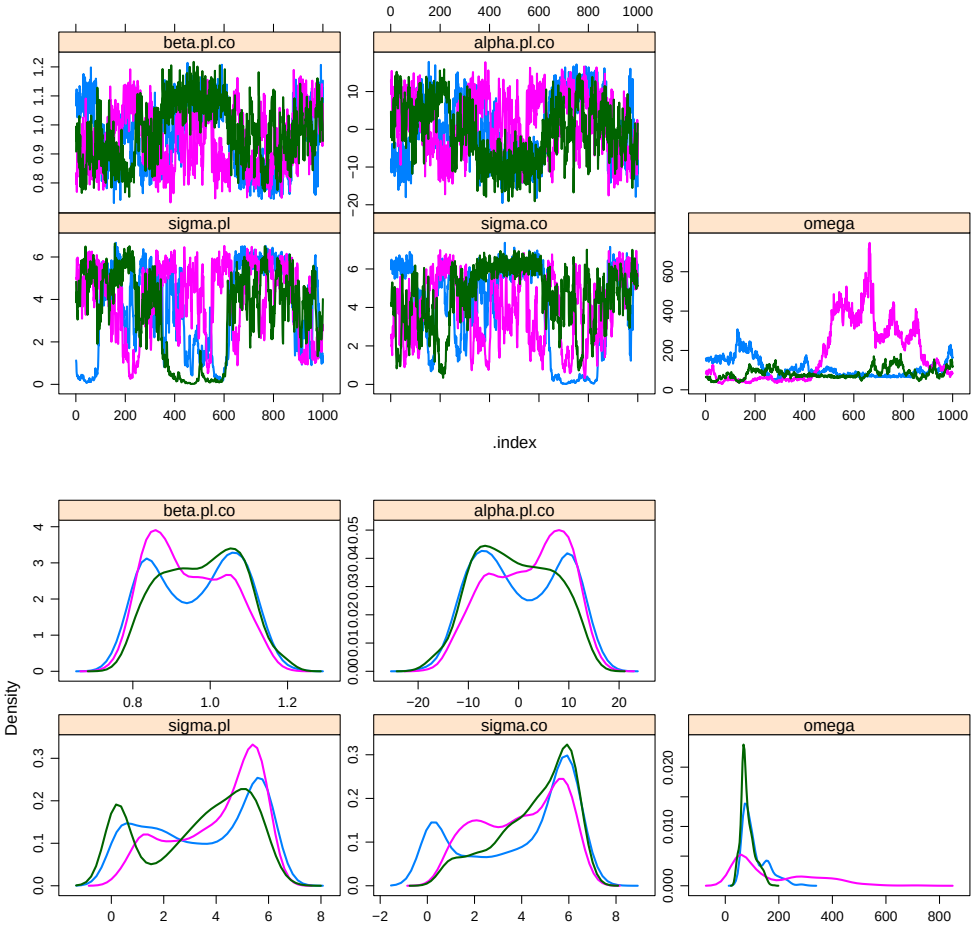

Figure 3.25: *Traces and densities of transformed parameters .*

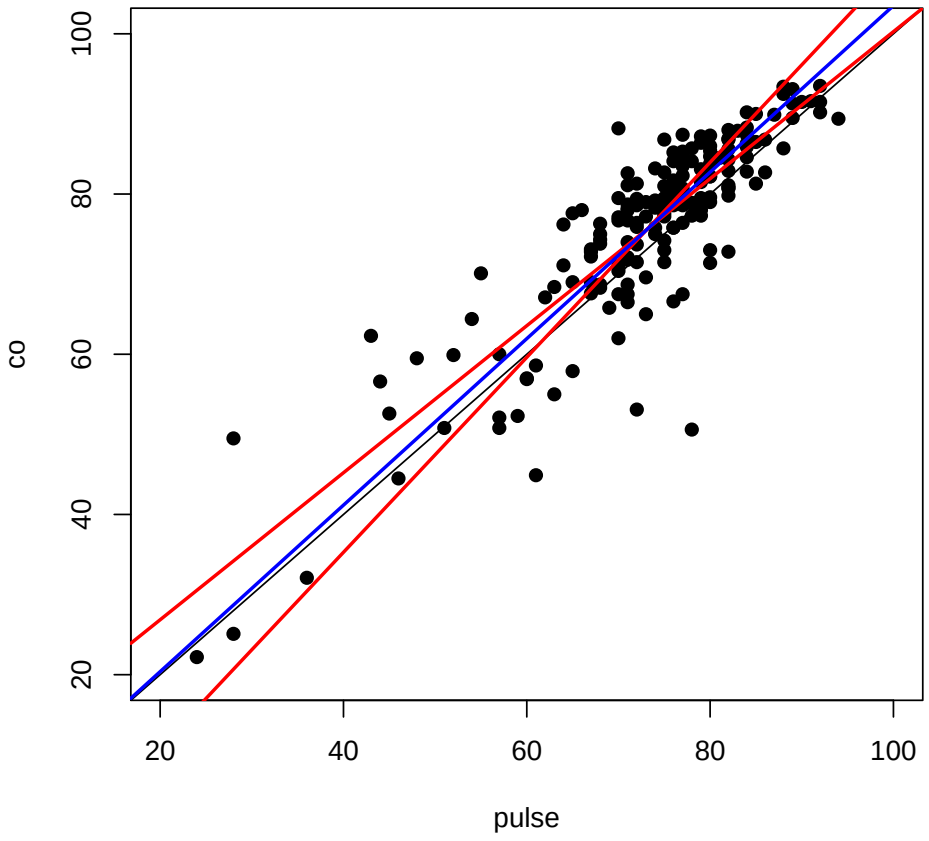

Figure 3.26: *Individual datapoints and traditional regression lines together with the line based on the posterior medians.*