9 Parameter estimation: Markov Chain Monte Carlo

In chapters 5 and 6 we found the posterior probability density function for one or two parameter problems either through an analytic approach or by computing the posterior on a grid. Here we will use the Metropolis algorithm described in chapter 8 to sample posteriors with more than two parameters. By way of example we will look at fitting curves (both linear and quadratic) $y = f(x) + \epsilon$, whereby we also infer the noise level in the data. We will see how we can use a mixture model to take account of outliers. I will finally show that it is conceptually straightforward in the Bayesian approach to accommodate uncertainties in both the x and y variables.

9.1 Fitting a straight line with unknown noise

In chapter 4 we looked at fitting a straight line by minimizing the sum of squares of the residuals of the data about that line. This is equivalent, for a Gaussian likelihood, to both maximum likelihood and to finding the maximum of the posterior with a uniform prior. A simple first-order propagation of errors (valid for small errors) allowed us to estimate uncertainties in the parameters and in the predictions.

If we don't just want to find the maximum, if we have more prior information, or if the errors are not small, we will want to properly characterise the distribution over the parameters by finding the posterior PDF. This will tell us more about the solutions than just a local maximum of the likelihood. We use the Monte Carlo methods discussed in the previous chapter to sample the PDF.

Suppose we have a two-dimensional set of N points $\{x_i, y_i\}$. The model M predicts the values of y as being

$$\begin{aligned} y &= f(x) + \epsilon \quad \text{where} \\ f(x) &= b_0 + b_1 x \end{aligned} \tag{9.1}$$

which is a straight line with parameters b_0 (intercept) and b_1 (gradient). $f(x)$ is the generative model: it gives the noise-free predictions of the data given the parameters. The residuals $\epsilon = y - f(x)$ are modelled as a zero-mean Gaussian random variable with standard deviation σ, i.e. $\epsilon \sim \mathcal{N}(0, \sigma)$. This is the *noise model*. Assuming the $\{x\}$ are noise free, this tells us that the likelihood is

$$P(y_i \,|\, x_i, \theta, M) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{[y_i - f(x_i; b_0, b_1)]^2}{2\sigma^2}\right] \tag{9.2}$$

where $\theta = (b_0, b_1, \sigma)$ are the parameters of the model. It may come as a surprise that we will try to infer the uncertainty in the data points from the data. Yet σ is as a model parameter just like the others. Although the x values are supplied with the data, we assume them to be fixed: they are not described by a measurement model. Thus the data are $D = \{y_i\}$. Assuming that the various y measurements are independent, the log likelihood for all N data points is

$$\ln P(\{y_i\} \,|\, \{x_i\}, \theta, M) = \sum_{i=1}^{N} \ln P(y_i \,|\, x_i, \theta, M). \tag{9.3}$$

From now on I will drop the conditioning both on $\{x_i\}$ for the sake of brevity, and on M because we only consider a single model. In general none of the three parameters is known in advance, so we want to infer their posterior PDF from the data, which is given as usual by

$$P(\theta \,|\, D) \propto P(D \,|\, \theta) P(\theta). \tag{9.4}$$

As we will be sampling from the posterior we do not need to compute the normalization constant.

Given a set of data, the procedure to compute the posterior is as follows:

(1) define the prior PDF over the parameters. I will use plausible yet convenient priors, and I will make use of a variable transformation;
(2) define the covariance matrix of the proposal distribution. I will use a diagonal, multivariate Gaussian;
(3) define the starting point (initialization) of the MCMC;
(4) define the number of burn-in iterations and the number of sampling iterations.

Once we have run the MCMC we perform the following analyses:

(5) thin the chains (see section 8.5.2);
(6) plot the chains and the one-dimensional marginal posterior PDFs over the parameters. I do the latter via kernel density estimation (see section 7.2.2);
(7) plot the two-dimensional posterior distributions of all three pairs of parameters, simply by plotting the samples (we could do two-dimensional kernel density estimation instead). I do this to look for correlations between the parameters;
(8) calculate the maximum a posteriori (MAP; see section 4.4.4) values of the model parameters from the MCMC chains, calculate and plot the resulting model, and compare to the original data;
(9) calculate the predictive posterior distribution over y at a new data point (following the approach described in section 8.2.4).

Note that because we have samples drawn from the posterior, we don't need the actual values of the posterior density in order to plot the posteriors. We likewise don't have to do any integration to get the one-dimensional marginal distributions (see point number 1 at the end of section 8.3).

I should point out that, with a suitable choice of priors, this problem has an analytic solution for the posterior. But in the general case we need to sample the posterior.

9.1.1 Priors

I adopt the following priors on the three model parameters.

- Intercept, b_0: $P(b_0) = \mathcal{N}(\mu, s)$, a Gaussian with mean μ and standard deviation s. One can estimate the two parameters of this prior by inspection of the general properties of the data.[1] This might be difficult in practice if the data are far from $x = 0$, in which case it would be better to centre the data. We will do this in section 9.1.6, but for now we use the data as they come.
- Gradient, b_1: We can write the gradient as $b_1 = \tan\alpha$, where α (in radians) is the angle between the horizontal and the model line. I assume that we have no prior knowledge of the slope of the line (not even its sign). This means we should use a uniform distribution in α, $P(\alpha) = 1/2\pi$ (or $1/\pi$, as a line is invariant under a rotation of π radians; but as we don't need the normalization constant this consideration is irrelevant). In contrast, a uniform distribution over the gradient would assign much less probability to lines with small angles than to lines with large ones.[2] It is preferable to use α as a parameter in an MCMC algorithm. With finite characteristic step sizes, a sampling over b_1 can never move the line from large positive to large negative gradients, because it cannot step over $b_1 = \pm\infty$.
- Standard deviation, σ: We argued in section 5.3.1 that, in the absence of any other information, a scale parameter such as the standard deviation of a Gaussian should be assigned a Jeffreys prior, $P(\sigma) \propto \log\sigma$. This also prevents σ from becoming negative. This is an improper prior, but this is often not a problem for determining the unnormalized posterior.

Given these priors, my model parameters are now $(b_0, \alpha, \log\sigma)$. These are the parameters that the Monte Carlo algorithm will sample over. The prior distributions are likewise defined over the parameters, as Gaussian, uniform, and uniform respectively. I therefore do not need to include the Jacobian mentioned in section 8.5.4 into the Metropolis algorithm. That would only be needed if the prior was specified as a density function over a different function of the parameters than that which is used by the sampling algorithm.

9.1.2 Sampling the posterior

The R code in section 9.1.4 below performs all of the above-mentioned tasks using some simulated data. I draw ten data points at random between $x = 0$ and $x = 10$, evaluate the straight line model (equation 9.1) with $b_0 = 0$ and $b_1 = 1$, then add zero mean Gaussian noise with standard deviation $\sigma = 1$ to produce the y values. The data are plotted in figure 9.1. From inspection of the data I assign mean zero and a generous standard deviation of two to the Gaussian prior on the intercept b_0. The priors on α and $\log\sigma$ have no parameters. The priors are set in the file `linearmodel_functions.R`. I initialize the MCMC using

[1] The parameters of a prior are sometimes called *hyperparameters* to distinguish them from the parameters of the model.

[2] You can easily show, using the method in section 1.9.1, that a uniform distribution over the angle α corresponds to a Cauchy distribution over the slope b_1.

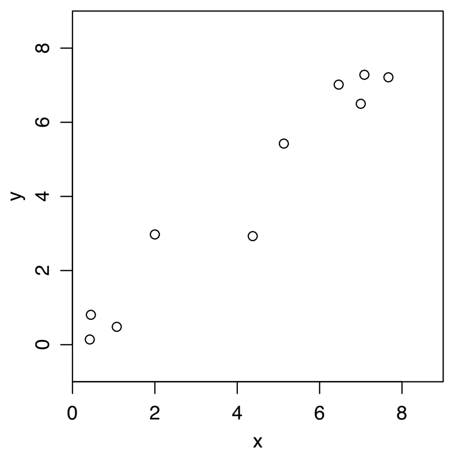

Fig. 9.1 Data used for the straight line fitting. They have been drawn at fixed x values from a straight line with $b_0 = 0$ and $b_1 = 1$ (equation 9.1), to which zero mean Gaussian noise with standard deviation 1 has been added.

values of the parameters that are intentionally far from being a good model – $b_0, \alpha, \log\sigma$ equal to $2, \pi/8, \log 3$ respectively – so that we can see how well/quickly the MCMC sampler finds regions of high probability density. The step sizes I set to $0.1, 0.02, 0.1$ for the three parameters (respectively), which intuitively seem to be small compared to the likely precision with which we can determine the parameters given these data.

We can now use MCMC to sample the posterior. Remember that it is sufficient if we use the unnormalized posterior – the product of likelihood and prior – in the MCMC, because we will draw samples in the same relative frequency whether it is normalized or not (see section 8.1). However, the posterior must be normaliz*able*. That is, it must be a proper distribution. If the posterior were improper we could still run an MCMC and we would still get a bunch of samples. But no finite number of samples would be representative of an entire improper distribution. If the prior is improper then in practice the likelihood will often ensure that the posterior is a proper distribution, but this is not guaranteed.

In contrast to the posterior, it is essential that the likelihood be normalized (not just normalizable). This is because it is a PDF over the data, so its normalization constant will generally be a function of the parameters we are sampling.

I run the MCMC for 50 000 iterations without burn-in. The average acceptance rate is around 0.56. I apply a thinning factor of 25, i.e. I retain only every twentyfifth sample in order to reduce the autocorrelation in the chain (see section 8.5.2). All results and plots that follow are based on the remaining 2000 samples. These are stored in the matrix `postSamp[,]`, for which the five columns are the log prior, log likelihood, b_0, α, and $\log\sigma$ respectively, with one row per iteration. The chains are shown in the left column of figure 9.2. We see how the sampler quickly moves to a new region of parameter space in the first few iterations. We should strictly remove these from the posterior estimates (i.e. use a burn-in). After this the search looks reasonably stable, and the chains look reasonable.

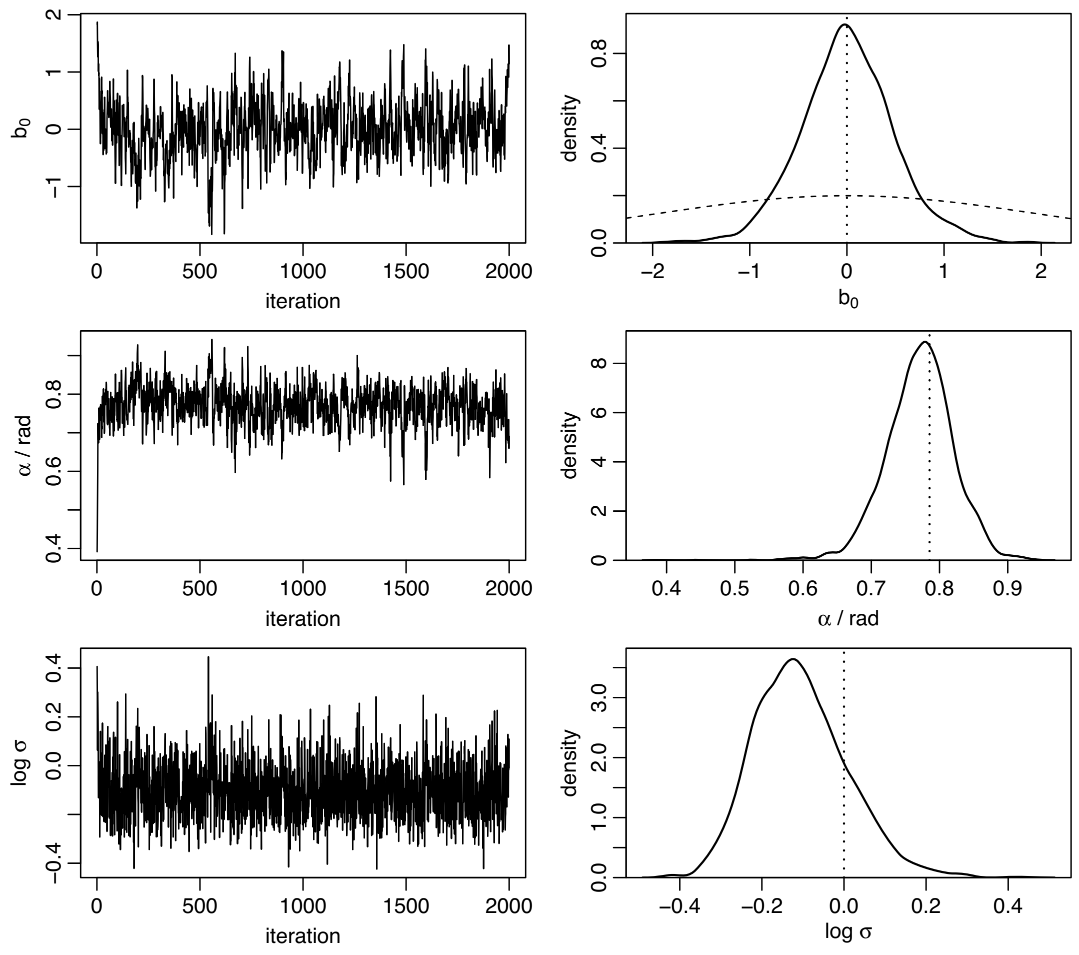

Fig. 9.2 MCMC chains (left columns) and resulting marginal posterior PDFs (right columns) for the straight line fitting problem. These one-dimensional posteriors have been computed by a kernel density estimate of the samples. The vertical dotted lines indicate the true parameters. These are only for reference: the parameters that best fit the noisy data are not necessarily the same as those that would best fit noise-free data. The curved dashed line in the panel for b_0 shows the prior distribution. The other priors are uniform.

The joint posterior distribution is the three-dimensional distribution over the MCMC samples. The one-dimensional marginalized distributions are obtained by making a density estimation of the samples for each parameter. These are shown in the right column of figure 9.2. Using the set of samples we can compute various statistics, for example the maximum or mean of the posterior as a single best estimate. In the present example the maximum is at $(b_0, \alpha, \log \sigma) = (0.036, 0.77, -0.19)$. This is the global maximum of the joint three-

Table 9.1 Summary of the covariance of the posterior PDF for the line-fitting problem. The leading diagonal gives the standard deviation of the three parameters. The off-diagonal elements give the correlation coefficients.

	b_0	α	$\log \sigma$
b_0	0.48		
α	−0.83	0.050	
$\log \sigma$	0.038	−0.073	0.11

dimensional posterior. This is not necessarily equal to the maxima of the corresponding one-dimensional marginalized distributions (although in this case they are very close). The mean of the posterior is $(b_0, \alpha, \log \sigma) = (0.0042, 0.77, -0.11)$. If we want to find the mean of the posterior over the original model parameters – (b_0, b_1, σ) – then we must transform the individual samples first and then compute the statistic (and not vice versa). This is because the mean is generally not invariant under a parameter transformation (see section 4.4.4). As the set of samples is representative of the distribution, transforming them essentially transforms the distribution. To find the mean of the transformed samples we do

```
mean(tan(postSamp[,4])) # transform alpha to b_1
mean(10^(postSamp[,5])) # transform log10(sigma) to sigma
```

which gives $(b_0, b_1, \sigma) = (0.0042, 0.98, 0.81)$. As a measure of the precision of these estimates we can calculate their standard deviations as well as the correlations between the parameters. These are shown in table 9.1. We can also inspect the correlations by plotting the parameter samples in two dimensions against each other. This is shown in figure 9.3 and confirms the values for the correlation coefficients in table 9.1. The strong negative correlation between b_0 and α arises because if we increase the slope of the line we need to decrease the intercept in order to retain a similar quality of fit for these data.

Having sampled the posterior we could now summarize it with, for example, the mean or maximum values, and see what this model looks like in the data space. This is shown in figure 9.4 using the maximum.[3] In this case the model at the mean of the parameters is almost identical, as is the least squares fit. This is not surprising, because I adopted broad priors on the parameters and the data are relatively informative. Note, however, that the least squares fit does not infer σ.

9.1.3 Posterior predictive distribution

Once we have decided on the "best" values for the model parameters (e.g. the maximum of the posterior), we could use these in the model to predict the value of y at any speci-

[3] Peek ahead to the right panel of figure 9.8 if you want to see a set of models drawn from the posterior, as opposed to just this single model at the mode.

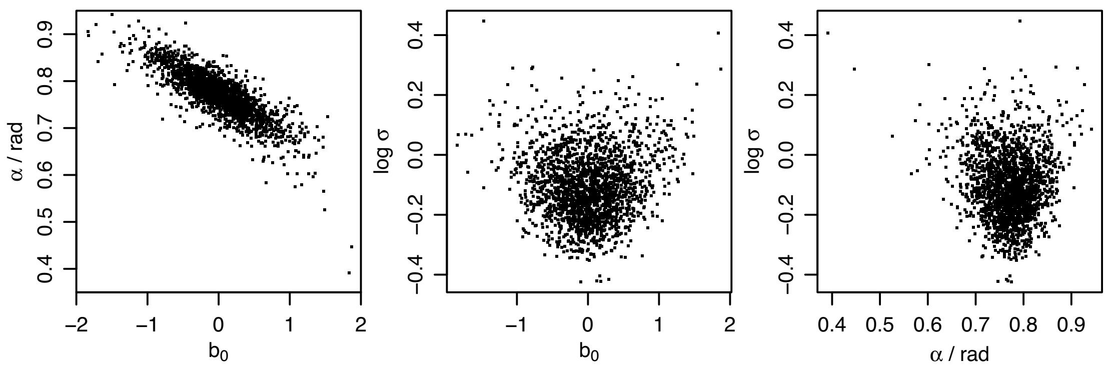

Fig. 9.3 The MCMC samples from the posterior PDF of the straight line model for each parameter, plotted pairwise to show the correlations.

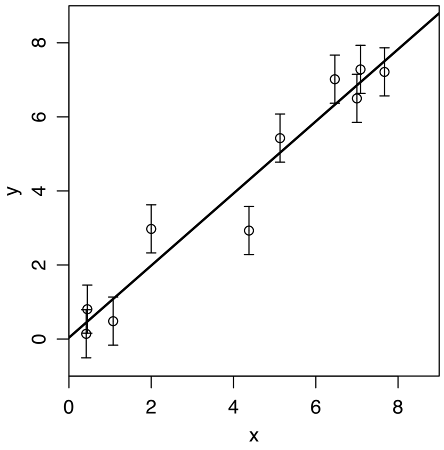

Fig. 9.4 The open circles are the data from figure 9.1. The straight line and error bars show the model obtained with the parameters set to the maximum of the posterior PDF (MAP), which has values $b_0 = 0.036$, $b_1 = 0.97$, $\sigma = 0.65$. The standard deviations in these and the correlations between them are listed in table 9.1.

fied x, call it x_p. We might further characterise the uncertainty in this prediction using the likelihood (equation 9.2). But the problem with this approach is that it gives a prediction and uncertainty estimate conditioned on a *single* set of parameters, thereby ignoring the uncertainty in the parameters reflected by the finite width of the posterior. How can we accommodate this? We saw in section 8.2.4 that the rules of probability lead us to incorporate uncertainties in parameters by marginalizing over them. This gave us the posterior predictive distribution

$$P(y_p \,|\, x_p, D) = \int \underbrace{P(y_p \,|\, x_p, \theta)}_{\text{likelihood}} \; \underbrace{P(\theta \,|\, D)}_{\text{posterior}} \, d\theta. \tag{9.5}$$

This distribution is computed in the code below in two ways. The "direct" approach is to explicitly evaluate $P(y_p \,|\, x_p, D)$ over a grid $\{y_p\}$, which is called `ycand` in the code. At a fixed value of y_p we take our set of N_s posterior samples $\{\theta_l\}$ (obtained by MCMC), calculate the likelihood at each of these, and then average these likelihoods, i.e.

$$P(y_p \,|\, x_p, D) \simeq \frac{1}{N_s} \sum_{l=1}^{N_s} P(y_p \,|\, x_p, \theta_l) \tag{9.6}$$

which was point number 4 at the end of section 8.3. Each of these likelihood calculations requires an evaluation of the generative model, as this gives the mean of the likelihood (see equation 9.2). We repeat this for each value in $\{y_p\}$ and plot these probability densities (called `ycandPDF` in the code) vs y_p. Equation 9.6 makes it clear that the posterior predictive distribution is a posterior-weighted average of the predictions (the likelihood) made at each θ. While this approach is accurate, it is slow, because it involves many likelihood evaluations for each value of y_p.

An alternative, "indirect" approach is to sample the joint distribution $P(y_p, \theta \,|\, x_p, D)$, and then to marginalize over θ. This helps because we can factorize the joint distribution as

$$P(y_p, \theta \,|\, x_p, D) = P(y_p \,|\, x_p, \theta) P(\theta \,|\, D). \tag{9.7}$$

whereby variables could be removed because of conditional independence (as explained immediately after equation 8.7). Each of the two PDFs on the right side can be represented by samples drawn from them. The second term is the posterior PDF; we already obtained the set of samples $\{\theta_l\}$ from this with the MCMC. The first term is the likelihood. We now sample this once (obtain a single value of y_p) for each value in $\{\theta_l\}$. As the likelihood is a univariate Gaussian (equation 9.2), it may be sampled using a standard function (`rnorm` in R). Its mean is the evaluation of the straight line at $(b_0, b_1)_l$, and its standard deviation is σ_l. Doing this for all N_s posterior samples gives a set of N_s predictions $\{y_p\}$ (called `likeSamp` in the code). The implementation in R is shorter than this explanation:

```
likeSamp <- rnorm(n=length(modPred), mean=modPred, sd=10^postSamp[,5])
```

where `modPred` (of length N_s) is the evaluations of the straight line at the posterior samples. We now have samples of θ and y_p. We marginalize their joint distribution simply by ignoring the θ, to give the required distribution $P(y_p \,|\, x_p, D)$. We then use density estimation to compute and plot their distribution (`likeDen` in the code).

In both the direct and indirect approaches we are effectively making predictions of y_p using all possible models (lines), and then weighting them by their posterior probability density computed from the data.

The left panel of figure 9.5 shows the posterior predictive distribution for $x_p = 6$ computed by both methods. We can take the maximum of $P(y_p \,|\, x_p, D)$ as the estimate of y_p, and use the 15.9% and 84.1% quantiles of this distribution as a measure of the (asymmetric) precision. These would correspond to the mode plus/minus 1 standard deviation if the posterior predictive distribution were Gaussian. Doing this we get $y_p = 5.86^{+0.84}_{-0.84}$ computed by the direct method (and $y_p = 5.80^{+0.92}_{-0.80}$ by the indirect method). This is shown in relation to the data in the right panel of figure 9.5. This value of x_p lies within the range

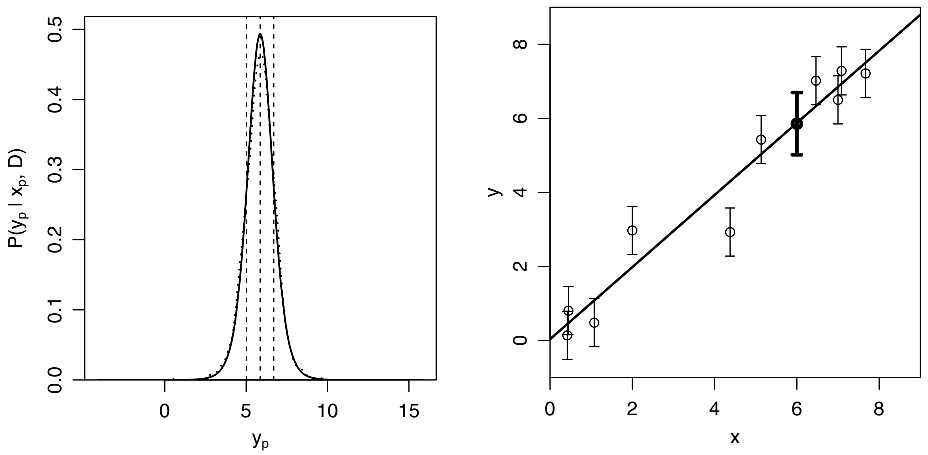

Fig. 9.5 The posterior predictive distribution $P(y_p\,|\,x_p, D)$ for $x_p = 6$ (left). The solid line is for the direct method, and the (barely distinguishable) dotted line is for the indirect method. Vertical dashed lines indicate the maximum as well as the 15.9% and 84.1% quantiles (from the direct method). The right plot shows this maximum as a thick point and these two quantiles as a thick error bar, plotted over the data and MAP model prediction as in figure 9.4.

of the original data. How does this method fare when we predict much further from the data (extrapolation)? An example is given in figure 9.6 for $x_p = 25$. Our predicted values are now $y_p = 24.4^{+2.1}_{-2.0}$ (and $y_p = 24.1^{+2.5}_{-1.7}$ with the indirect method). The posterior is necessarily wider in this case, because a given uncertainty in the gradient is amplified to a larger uncertainty in y_p the further the prediction is made from the data (think of the line as a "lever arm" pivoting about its intercept). Recall that the standard deviation of a prediction arising from the least squares fit in the case of known σ – derived in section 4.1.2 – showed a similar behaviour.

9.1.4 R code for fitting a straight line

The analysis described above is all done with the R script `linearmodel_posterior.R`, with explanations provided as comments in the code. The code looks long, but a reasonable chunk of it is actually concerned with plotting and analysing the results. The code initially sources two other files. The first file, `linearmodel_functions.R`, defines functions that compute the (logarithm of the) prior, likelihood, and posterior. This is listed below. The second file is the Metropolis algorithm listed in section 8.6.2. The rest of the code is then executed.

To better appreciate how MCMC works in this example, I recommend you experiment with the code, and change in particular:

- the initialization of the parameters;
- the parameter step sizes;

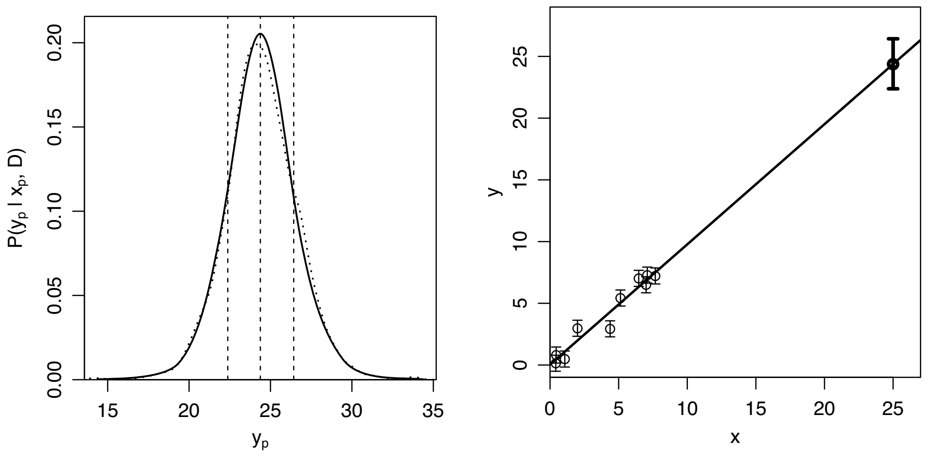

Fig. 9.6 As figure 9.5, but for a prediction at $x_p = 25$.

- the number of iterations and the length of the burn-in;
- the value of the standard deviation of the prior on b_0;
- the prior on b_0 from a Gaussian to an improper uniform prior. Do this by setting `b0Prior` to unity in the function `logprior.linearmodel`;
- the amount of data. Try both more data points (e.g. 100) and fewer, including – before you read section 9.1.5 – just three, two, and one data points.

R file: `linearmodel_posterior.R`

```
##### Bayesian inference of a 3-parameter linear model to 2D data

library(gplots) # for plotCI
source("metropolis.R")
source("linearmodel_functions.R") # provides logpost.linearmodel

########## Define true model and simulate experimental data from it

set.seed(50)
Ndat <- 10
x <- sort(runif(Ndat, 0, 10))
sigTrue <- 1
modMat <- c(0,1) # 1 x P vector: coefficients, b_p, of sum_{p=0} b_p*x^p
y <- cbind(1,x) %*% as.matrix(modMat) + rnorm(Ndat, 0, sigTrue)
# Dimensions in the above: [Ndat x 1] = [Ndat x P] %*% [P x 1] + [Ndat]
# cbind does the logical thing when combining a scalar and vector,
# then do vector addition
y <- drop(y) # converts into a vector
pdf("linearmodel_data.pdf", width=4, height=4)
par(mfrow=c(1,1), mar=c(3.5,3.5,0.5,0.5), oma=0.1*c(1,1,1,1),
    mgp=c(2.0,0.8,0), cex=1.0)
plot(x, y, xlim=c(0,9), ylim=c(-1,9), xaxs="i", yaxs="i")
#abline(a=modMat[1], b=modMat[2], col="red") # true model
dev.off()
```

```
# True parameters, transformed to conform with model to be used below
thetaTrue <- c(modMat[1], atan(modMat[2]), log10(sigTrue))
obsdata <- data.frame(cbind(x,y)) # columns must be named "x" and "y"
rm(x,y)

### Define model and infer the posterior PDF over its parameters

# Model to infer: linear regression with Gaussian noise
# Parameters: intercept b_0, gradient b_1; Gaussian noise sigma, ysig.
# MCMC works on: theta=c(b_0, alpha=tan(b_1), log10(ysig)), a 1x3 vector.
# Prior PDFs:
# b_0:         N(mean=m, sd=s); m,s estimated from global properties of data
# alpha:       Uniform (0 to 2pi)
# log10(ysig): Uniform (improper)

# Define covariance matrix of MCMC sampling PDF:
# sigma=c(b_0, alpha, log10(ysig))
sampleCov <- diag(c(0.1, 0.02, 0.1)^2)
# Set starting point
thetaInit <- c(2, pi/8, log10(3))
# Run the MCMC to get samples from the posterior PDF
set.seed(150)
allSamp <- metrop(func=logpost.linearmodel, thetaInit=thetaInit, Nburnin=0,
                  Nsamp=5e4, sampleCov=sampleCov, verbose=1e3,
                  obsdata=obsdata)
# 10^(allSamp[,1]+allSamp[,2]) is the unnormalized posterior at each sample
thinSel  <- seq(from=1, to=nrow(allSamp), by=25) # thin by factor 25
postSamp <- allSamp[thinSel,]

# Plot MCMC chains and use density estimation to plot 1D posterior PDFs.
# We don't need to do any explicit marginalization to get the 1D PDFs.
pdf("linearmodel_mcmc.pdf", width=8, height=7)
par(mfrow=c(3,2), mar=c(3.0,3.5,0.5,0.5), oma=0.5*c(1,1,1,1),
    mgp=c(1.8,0.6,0), cex=0.9)
parname <- c(expression(b[0]), expression(paste(alpha, " / rad")),
             expression(paste(log, " ", sigma)))
for(j in 3:5) { # columns of postSamp
  plot(1:nrow(postSamp), postSamp[,j], type="l",
       xlab="iteration", ylab=parname[j-2])
  postDen <- density(postSamp[,j], n=2^10)
  plot(postDen$x, postDen$y, type="l", lwd=1.5, yaxs="i",
       ylim=1.05*c(0,max(postDen$y)), xlab=parname[j-2], ylab="density")
  abline(v=thetaTrue[j-2], lwd=1.5, lty=3)
  if(j==3) { # overplot prior
    b0Val <- seq(from=-8, to=8, by=0.01)
    lines(b0Val, dnorm(b0Val, mean=0, sd=2), lty=2)
  }
}
dev.off()

# Plot gradient and intercept of samples in 2D.
# Fix range for b_0 vs alpha to enable comparison to centered data case
pdf("linearmodel_parameter_correlations.pdf", width=6, height=2)
par(mfrow=c(1,3), mgp=c(2.0,0.8,0), mar=c(3.0,3.0,0.5,0.5),
    oma=0.1*c(1,1,1,1))
plot(postSamp[,3], postSamp[,4], xlab=parname[1], ylab=parname[2],
     pch=".", xlim=c(-2,2), ylim=c(0.35,0.95), xaxs="i", yaxs="i")
plot(postSamp[,3], postSamp[,5], xlab=parname[1], ylab=parname[3], pch=".")
plot(postSamp[,4], postSamp[,5], xlab=parname[2], ylab=parname[3], pch=".")
dev.off()

# Find MAP and mean solutions.
# MAP is not the peak in each 1D PDF, but the peak of the 3D PDF.
# Mean is easy, as samples were drawn from the (unnormalized) posterior.
posMAP    <- which.max(postSamp[,1]+postSamp[,2])
thetaMAP  <- postSamp[posMAP, 3:5]
thetaMean <- apply(postSamp[,3:5], 2, mean) # Monte Carlo integration
cov(postSamp[, 3:5]) # covariance
cor(postSamp[, 3:5]) # correlation

# Overplot these solutions with original data
pdf("linearmodel_fits.pdf", width=4, height=4)
par(mfrow=c(1,1), mar=c(3.5,3.5,0.5,0.5), oma=0.1*c(1,1,1,1),
    mgp=c(2.0,0.8,0), cex=1.0)
plotCI(obsdata$x, obsdata$y, xlim=c(0,9), ylim=c(-1,9), xaxs="i", yaxs="i",
       xlab="x", ylab="y", uiw=10^thetaMAP[3], gap=0)
abline(a=thetaMAP[1], b=tan(thetaMAP[2]), lw=2)     # MAP  model
#abline(a=modMat[1],   b=modMat[2], col="red", lw=2) # true model
# Compare this with the result from ML estimation from lm()
#abline(lm(obsdata$y ~ obsdata$x), col="black", lty=2)
dev.off()

### Make prediction: determine PDF(ycand | xnew, obsdata)

# Model and likelihood used here must be consistent with logpost.linearmodel

# Example 1
xnew <- 6
xlim <- c( 0,9) # xlim and ylim for plotting only
ylim <- c(-1,9)
# Example 2
#xnew <- 25
#xlim <- c( 0,27)
#ylim <- c(-1,29)

# Evaluate generative model at posterior samples (from MCMC).
# Dimensions in matrix multiplication: [Nsamp x 1] = [Nsamp x P] %*% [P x 1]
modPred <- cbind(postSamp[,3], tan(postSamp[,4])) %*% t(cbind(1,xnew))

# Direct method
# ycand must span full range of likelihood and posterior
dy    <- 0.01
ymid  <- thetaMAP[1] + xnew*tan(thetaMAP[2]) # to center choice of ycand
ycand <- seq(ymid-10, ymid+10, dy) # uniform grid of y with step size dy
ycandPDF <- vector(mode="numeric", length=length(ycand))
for(k in 1:length(ycand)) {
  like <- dnorm(ycand[k], mean=modPred, sd=10^postSamp[,5]) # [Nsamp x 1]
  ycandPDF[k] <- mean(like) # integration by rectangle rule. Gives a scalar
}
# Note that ycandPDF[k] is normalized, i.e. sum(dy*ycandPDF)=1.
# Find peak and approximate confidence intervals at 1sigma on either side
peak.ind  <- which.max(ycandPDF)
```

```
lower.ind <- max( which(cumsum(dy*ycandPDF) < pnorm(-1)) )
upper.ind <- min( which(cumsum(dy*ycandPDF) > pnorm(+1)) )
yPredDirect <- ycand[c(peak.ind, lower.ind, upper.ind)]

# Indirect method. likeSamp is [Nsamp x 1]
likeSamp <- rnorm(n=length(modPred), mean=modPred, sd=10^postSamp[,5])
likeDen  <- density(likeSamp, n=2^10)
# Find peak and confidence intervals
yPredIndirect <- c(likeDen$x[which.max(likeDen$y)], quantile(likeSamp,
                          probs=c(pnorm(-1), pnorm(+1)), names=FALSE))

# Plot the predictive posterior distribution
pdf("linearmodel_prediction6_PDF.pdf", width=4, height=4)
par(mfrow=c(1,1), mar=c(3.0,3.5,0.5,0.5), oma=0.5*c(1,1,1,1),
    mgp=c(2.2,0.8,0), cex=1.0)
plot(ycand, ycandPDF, type="l", lwd=1.5, yaxs="i",
     ylim=1.05*c(0,max(ycandPDF)), xlab=expression(y[p]),
     ylab=expression(paste("P(", y[p], " | ", x[p], ", D)")))
abline(v=yPredDirect, lty=2)
# overplot result from the indirect method
lines(likeDen$x, likeDen$y, type="l", lty=3, lwd=1.5)
dev.off()

# Compare predictions between the two methods
rbind(yPredDirect, yPredIndirect)

# Overplot direct prediction with original data and the MAP model
pdf("linearmodel_prediction6_ondata.pdf", width=4, height=4)
par(mfrow=c(1,1), mar=c(3.5,3.5,0.5,0.5), oma=0.1*c(1,1,1,1),
    mgp=c(2.0,0.8,0), cex=1.0)
plotCI(obsdata$x, obsdata$y, xlim=xlim, ylim=ylim, xaxs="i", yaxs="i",
       uiw=10^thetaMAP[3], gap=0, xlab="x", ylab="y")
abline(a=thetaMAP[1], b=tan(thetaMAP[2]), lwd=2) # MAP  model
plotCI(xnew, ycand[peak.ind], li=ycand[lower.ind], ui=ycand[upper.ind],
  gap=0, add=TRUE, lwd=3)
dev.off()
```

R file: `linearmodel_functions.R`

```
##### Functions to evaluate the prior, likelihood, and posterior
##### for the linear model, plus to sample from the prior

# theta is vector of parameters
# obsdata is 2 column dataframe with names [x,y]
# The priors are hard-wired into the functions

# Return c(log10(prior), log10(likelihood)) (each generally unnormalized)
# of the linear model
logpost.linearmodel <- function(theta, obsdata) {
  logprior <- logprior.linearmodel(theta)
  if(is.finite(logprior)) { # only evaluate model if parameters are sensible
    return( c(logprior, loglike.linearmodel(theta, obsdata)) )
  } else {
    return( c(-Inf, -Inf) )
  }
}
```

```
# Return log10(likelihood) (a scalar) for parameters theta and obsdata
# dnorm(..., log=TRUE) returns log base e, so multiply by 1/ln(10) = 0.434
# to get log base 10
loglike.linearmodel <- function(theta, obsdata) {
  # convert alpha to b_1 and log10(ysig) to ysig
  theta[2] <- tan(theta[2])
  theta[3] <- 10^theta[3]
  modPred <- drop( theta[1:2] %*% t(cbind(1,obsdata$x)) )
  # Dimensions in mixed vector/matrix products: [Ndat] = [P] %*% [P x Ndat]
  logLike <- (1/log(10))*sum( dnorm(modPred - obsdata$y, mean=0,
                                    sd=theta[3], log=TRUE) )
  return(logLike)
}

# Return log10(unnormalized prior) (a scalar)
logprior.linearmodel <- function(theta) {
  b0Prior      <- dnorm(theta[1], mean=0, sd=2)
  alphaPrior   <- 1
  logysigPrior <- 1
  logPrior <- sum( log10(b0Prior), log10(alphaPrior), log10(logysigPrior) )
  return(logPrior)
}
```

9.1.5 Discussion

Why is the inferred straight line sometimes quite different from the true straight line?
Don't be mislead by the fact that with noisy data the inferred model is not identical to the true model. We can only work with the noisy data; we never have access to the true model. If we are "unlucky" that the data drawn are far from the true model, then we will infer a different straight line. For this reason I intentionally did not plot the true model in figure 9.1 and similar figures (but you can do so by uncommenting a line in the code). When we know the true model, we should find that the inferred model (e.g. maximum of the posterior) converges to the true one as we use more data.

How can we infer the errors bars as well as the line?
Because the data are noisy, they will not normally lie on a straight line. So it should be clear that if we guessed a value of σ that was small compared to the true (unknown) value, then the likelihood of the data for any given line would be small (as there would be a large spread in the data compared to our σ, putting us right down in the tails of the likelihood for most data points; see equation 9.2). Increasing our guess for σ slightly would increase the likelihood. You might think that an ever larger σ would always increase the likelihood for a given data point, because the Gaussian becomes wider. But don't forget that the Gaussian likelihood is normalized, so as it gets wider it also also gets lower (see figure 1.5). Mathematically, the exponential part of the Gaussian gets larger, but the $1/\sigma$ term in front gets smaller: as $\sigma \to \infty$, the likelihood tends to zero (for finite data). Thus for given data and a given line (set by the other two parameters), there must be an intermediate value of σ that maximizes the likelihood.

What happens if we have much more data?
As we saw in chapter 5, in particular figure 5.6, the more informative the data – which often corresponds to having more data – the narrower the likelihood becomes as a function of the parameters. The prior remains unchanged. The posterior thus becomes increasingly determined by the data and less dependent on the choice of prior.

How are the results affected by the choice of priors?
You can easily investigate this by editing the function `logprior.linearmodel`, and I encourage you to do so. In the set-up above I have used uniform priors on α and $\log\sigma$, which are reasonably uninformative, but I used a Gaussian prior on b_0. If we instead use a uniform improper prior on b_0, the resulting posteriors (with ten data points) are hardly any different: this Gaussian prior was already quite uninformative compared to these data. I have intentionally taken an abstract example here without a scientific context, so that we can concentrate on the mechanics of posterior inference using MCMC. In practice we would have some information that would influence our choice of priors. For example, we invariably have some idea of the scale of the measurement uncertainties. We could use this information to set the standard deviation of the prior on σ. I adopted here a prior that is uniform over α. Quite often we would have background information which tells us a priori that the slope must be constrained to a narrower region. For example, if y were the distance travelled by an electron in time x, then we know that the slope – the speed – cannot be larger than the speed of light. We could scale the range of standard distributions (such as the beta) to impose a variable prior with hard constraints, for example. Note that the uniform prior over α is not invariant with respect to a change of the measurement units of x and y. This may not be such a problem, however, as physical constraints on its values would be expressed in the same units.

What happens if we reduce the size of the data set to two points, or even just one?
Even with fewer data points than parameters, the likelihood function is still defined, so we still get a posterior PDF. The posterior is likely to become a lot wider, because with less data, the less well we can determine the parameters. But we still have a distribution from which we can estimate the mode of the parameters (and the mean if it is still a normalizable distribution). If you don't believe me, try it.[4] But do look carefully at the MCMC chains: they are likely to be poor – not in a steady state – when we have few data. If that is the case then they are not representative of the posterior. To remedy this you may need a lot more samples, a longer burn-in, more thinning, and/or a different covariance matrix for the proposal distribution.

But how, logically thinking, can we estimate three parameters at all given only one or two data points? Surely the solution is somehow "underdetermined"?
No, it's not. You're not only using the data in your inference. You also have a prior. The prior on α is uniform, but the prior on the intercept (b_0) is not. We used a Gaussian with

[4] The R code will work with just one data point. This is not immediately obvious, however, because when a vector or matrix only has one element, then R (sometimes) automatically converts it into a scalar, which may then be an incompatible data type for some calculation (e.g. a matrix multiplication).

mean zero and standard deviation 2. You can think of this as acting as a constraint on the model. Specifically, the prior on the intercept is like an additional fuzzy data point (y value) at $x = 0$. So when combined with a single data point, this will tell you something about the gradient. If we increased the width of the intercept prior, then the posterior over this parameter (and typically that over the gradient) would become broader. More fundamentally, the posterior represents all the information we have on the parameters. Even with little data the posterior constrains the parameters more than the prior. Note that the maximum likelihood (least squares) solution, as produced by `lm`, is not defined when there are fewer data points than parameters. The Bayesian solution, in contrast, always exists.

What if we have no data?
In that case the likelihood function is not defined (it doesn't even exist). The posterior is then just equal to the prior. Indeed, that's exactly what the prior is: our knowledge of the PDF over the parameters in the absence of data.[5]

9.1.6 Centring the data

In the above example I used a Gaussian prior $\mathcal{N}(\mu, s)$ on the intercept parameter b_0. I extrapolated the data by eye to the vertical axis in figure 9.1 in order to chose suitable values of μ and s. Extrapolating like this is generally cumbersome, especially if the data lie far from the origin. It is more convenient to first subtract the mean from the data, i.e. transform them to $x' = x - \overline{x}$ and $y' = y - \overline{y}$. This is called *centring* the data. We can then set $\mu = 0$ and s to a generous estimate of the possible range of the data at $y' = 0$. Estimating priors from the general properties of the data in this way is known as *empirical Bayes*.

As centring is just a linear shift of the data, it does not affect the priors we adopt for α and $\log \sigma$. If is simple to modify the R code in section 9.1.4 to deal with centred data:

- Redefine `obsdata`

```
xMean <- mean(x)
yMean <- mean(y)
obsdata <- data.frame(cbind(x=x-xMean,y=y-yMean))
```

- Set the prior on the intercept for the model of the centred data. In fact, $\mu = 0$ and $s = 2$ are still reasonable values, so you do not have to change anything.
- If you want to plot the true model remember that it is defined with an intercept b_0 in the original data space. In the centred data space the intercept is $b_0 - \bar{y} + b_1\overline{x}$.

Without making any further changes the code can be run again to produce the posterior samples. You will find that the chains and posteriors are almost identical to what we had before. But there is one important difference. Now that the data are centred, the gradient

[5] The R code will not work properly if you set `data` to be `NULL`. This is because the code does not accommodate the likelihood being undefined (although that would be easy to fix). The code actually does return a value, but it is not the value you want.

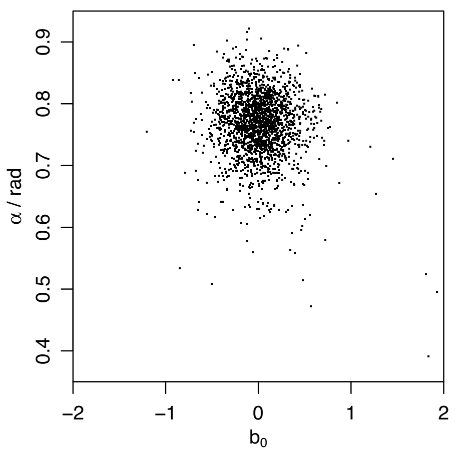

Fig. 9.7 The posterior MCMC samples for b_0 and α for the straight line model applied to the centred data. We see that the correlation between the parameters has been reduced by the centring.

and intercept are much less correlated. This can be seen from the plot of the samples in figure 9.7. Compare this to the case when we did not centre the data (the left panel of figure 9.3). Using centred data the correlation decreases in magnitude from -0.83 to -0.11. We saw when doing least squares fitting in section 4.1.1 that the uncertainties in the parameters also become decorrelated when $\overline{x} = 0$ (equation 4.17).

Figure 9.4 showed how one particular posterior model, the one corresponding to the maximum of the posterior PDF, appears in the data space. It is instructive to plot a sample of posterior models. This is shown in the right panel of figure 9.8. Each grey line corresponds to one set of parameters drawn from the posterior (although it does not show the σ parameter). The variation in intercepts and gradients reflects the finite width of the posterior PDF. We can compare this to a sample of models drawn from the prior, show in the left panel. The prior is uniform in α and has a small range of b_0, so the lines cover all position angles and pivot near to the origin. We see that these prior models cover a much broader range of the data space than the posterior models.[6] This again illustrates how the process of inference combines the prior with the likelihood to form the posterior: we update our prior knowledge by using the data. Here the data are informative, so our knowledge of the model parameters is improved.

The prior models we used with the original data were actually quite inappropriate because they were also lines of all angles pivoting around the origin, whereas the data were uncentred. We nonetheless managed to get good results (equally tight posteriors) because the data were informative, so the likelihood dominated the prior to make the posterior.

[6] Had I used an improper uniform prior on b_0 I would not have been able to draw from the prior for b_0 so I could not have made this plot.

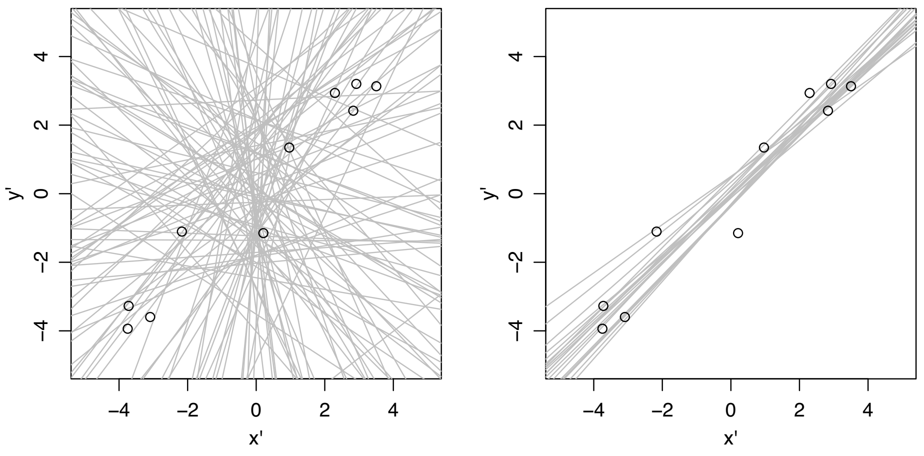

Fig. 9.8 Models drawn from the prior (left) and from the posterior (right) for the straight line model. The open circles show the centred data.

9.2 Fitting a quadratic curve with unknown noise

We can repeat the same procedure as in the previous section, but now for a quadratic generative model

$$y = f(x) + \epsilon \quad \text{where} \tag{9.8}$$

$$f(x) = b_0 + b_1 x + b_2 x^2 \tag{9.9}$$

and again $\epsilon \sim \mathcal{N}(0, \sigma)$. The R script `quadraticmodel_posterior.R` and the functions file `quadraticmodel_functions.R` (both available online only) are similar to the straight line model in the previous section, but now with a quadratic term added. I again use a Gaussian prior on b_0, transform b_1 to $\alpha = \arctan(b_1)$ and use a uniform prior on this, and transform σ to $\log \sigma$ and use an improper uniform prior on this too. To b_2 I apply a Gaussian prior.

In the code I draw 20 data points at random between $x = 0$ and $x = 10$, evaluate the quadratic model (equation 9.9) with $b_0 = 25$, $b_1 = -10$, and $b_2 = 1$, and add zero mean Gaussian noise with standard deviation $\sigma = 2$ to produce the y values. These simulated data are plotted in figure 9.9. For the Gaussian priors I use $\mathcal{N}(0, 10)$ for b_0 and $\mathcal{N}(0, 5)$ for b_2. These are quite broad and are adopted purely for illustration purposes. The priors on α and $\log \sigma$ are again uniform and have no parameters. The priors are set in `quadraticmodel_functions.R`.

For the four parameters b_0, α, b_2, and $\log \sigma$ I use step sizes (Gaussian standard deviations) in the MCMC of 0.1, 0.01, 0.01, and 0.01 respectively. In principle I can initialize the sampling anywhere, but it could then take an extremely large number of steps to locate the high density regions of the posterior. The more parameters we have, the more acute this problem becomes. It is far better to take a good guess at the values of the parameters. Here I

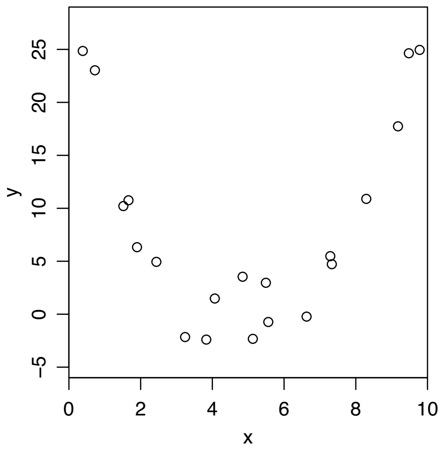

Fig. 9.9 Data used for the quadratic model fitting. They have been drawn from a quadratic curve at fixed x values with $b_0 = 25$, $b_1 = -10$, $b_2 = 1$ (equation 9.9), to which zero mean Gaussian noise with standard deviation 2 has been added.

do one better by finding the maximum likelihood (least squares) solution of the parameters b_0, α, and b_2 (using `lm`) and use these as the initial condition for the Markov chain. For the initial value of σ I use the root mean square of the residuals about this fit.

To achieve good chains we nonetheless need more iterations than in the straight line problem, on account of the higher complexity of the model (we have one more parameter). After a burn-in of 20 000 iterations, I sample for a further 200 000 iterations. To reduce the autocorrelation I use a thinning factor of 100. The resulting chains (for the 2000 samples retained) are shown in the left column of figure 9.10. We still see some evidence for autocorrelations. Yet the marginal posterior PDFs, show in the right column, are not very sensitive to this. Their shapes and summary statistics do not change much if we double or half the number of iterations or the degree of thinning.

If we take the parameter values at the maximum of the posterior, this gives a good model for the data: this is the curve and error bars shown figure 9.11. The mean of the parameter samples (not shown) gives an almost identical fit. We should be careful with the mean, however: it takes into account all samples, so it can be more affected if the sampling algorithm produces unrepresentative samples by getting stuck in an island of low probability.

Figure 9.12 shows the correlations between the samples. We see correlations between all parameters in the generative model, in particular a strong anticorrelation between α and b_2. This is not that surprising, because we can compensate for a small change in one of these parameters by a small change in the other, i.e. in order to retain more or less the same curve. There is barely any correlation between the inferred noise σ and the generative model parameters.

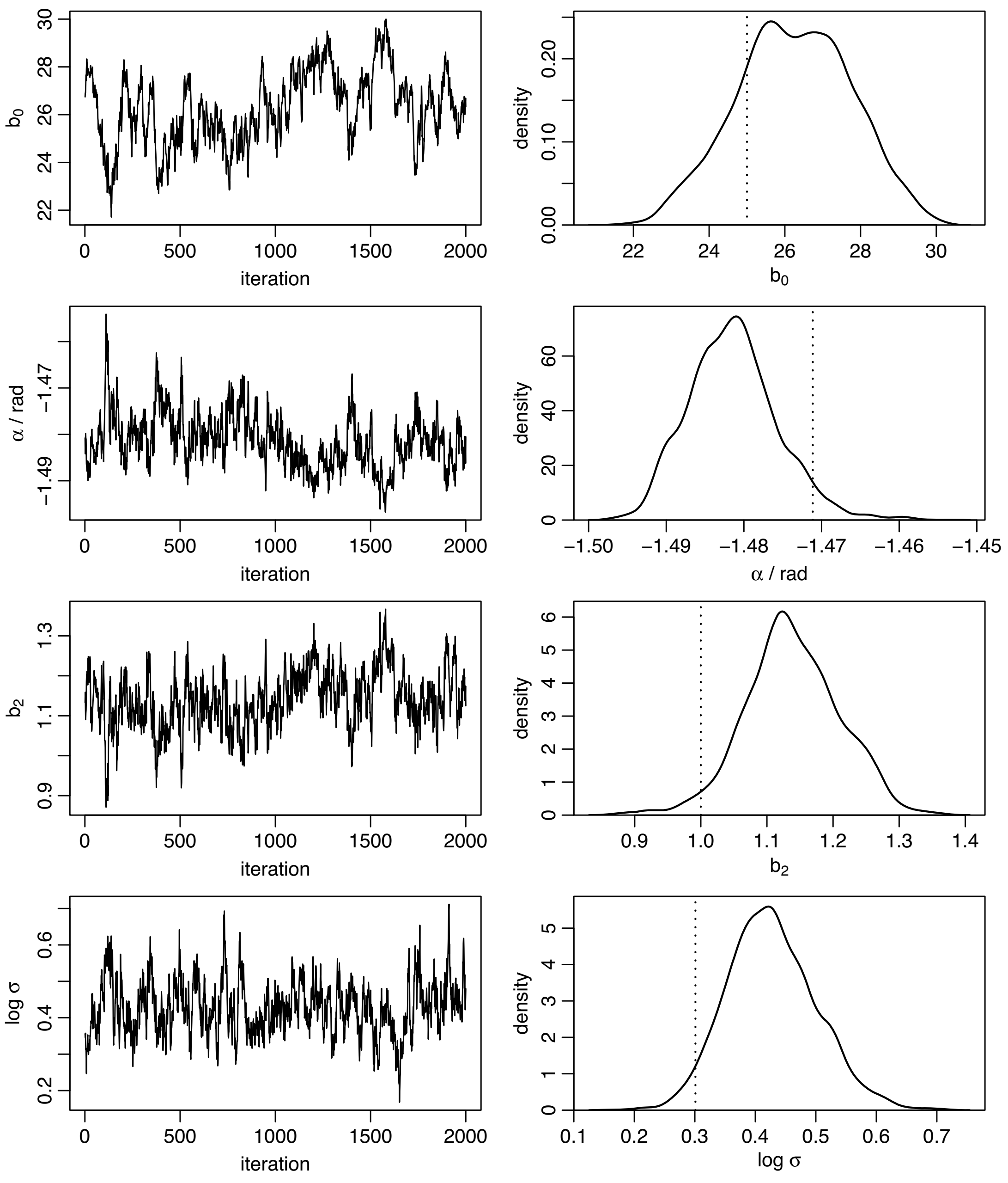

Fig. 9.10 MCMC chains (left columns) and resulting marginal posterior PDFs (right columns) for the quadratic model fitting problem. The one-dimensional posteriors have been computed by a kernel density estimate of the samples. The vertical dotted lines indicate the true parameters. The priors for b_0 and b_2 are Gaussian, with zero mean and standard deviation 10 and 5 respectively, so they are virtually flat over the range of the parameters plotted. The priors for the other two parameters are uniform.

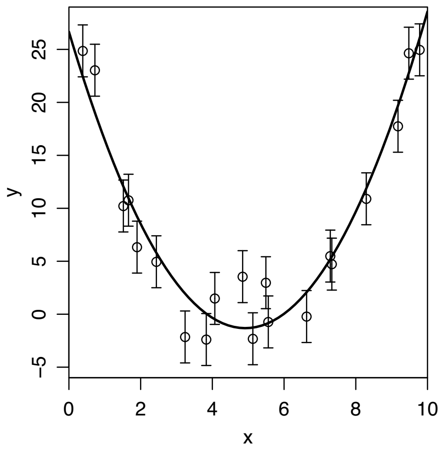

Fig. 9.11 The open circles are the data from figure 9.9. The curve and error bars show the model obtained with the parameters set to the maximum of the posterior PDF (MAP), which has values $b_0 = 26.7$, $b_1 = -11.4$, $b_2 = 1.16$, $\sigma = 2.45$.

9.3 A mixture model: fitting a straight line with an outlier model

In the straight line and quadratic model examples in the previous sections, we assumed that the residuals $y - f(x)$ followed a zero mean Gaussian distribution (the likelihood) with common, but unknown, standard deviation σ. But what if the data contains outliers, as data tend to? These could distort the line fit and/or increase the estimated value of σ. One could attempt to remove or manually down-weight outliers, but such approaches tend to be ad hoc and unstable. Could we instead deal with them consistently – probabilistically – in the modelling procedure?

Virtually by definition an outlier is a data point that we believe is not drawn from the adopted likelihood. The fact that it sticks out (rather than in, where we would not notice it) suggests it has come from a broader likelihood distribution. We could model outliers with a second Gaussian, but more convenient is a Cauchy distribution (section 1.4.7), because it has much broader tails. Its PDF for point (x, y) is

$$L_{\text{out}}(y) = \frac{1}{\pi w[1 + (\frac{y-f(x)}{w})^2]} \tag{9.10}$$

where w is the half-width at half-maximum, and we have set the mode equal to the model-predicted values (the outlier distribution is symmetric about this). We can then combine this with the main likelihood distribution

$$L_{\text{main}}(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{[y - f(x)]^2}{2\sigma^2}\right] \tag{9.11}$$

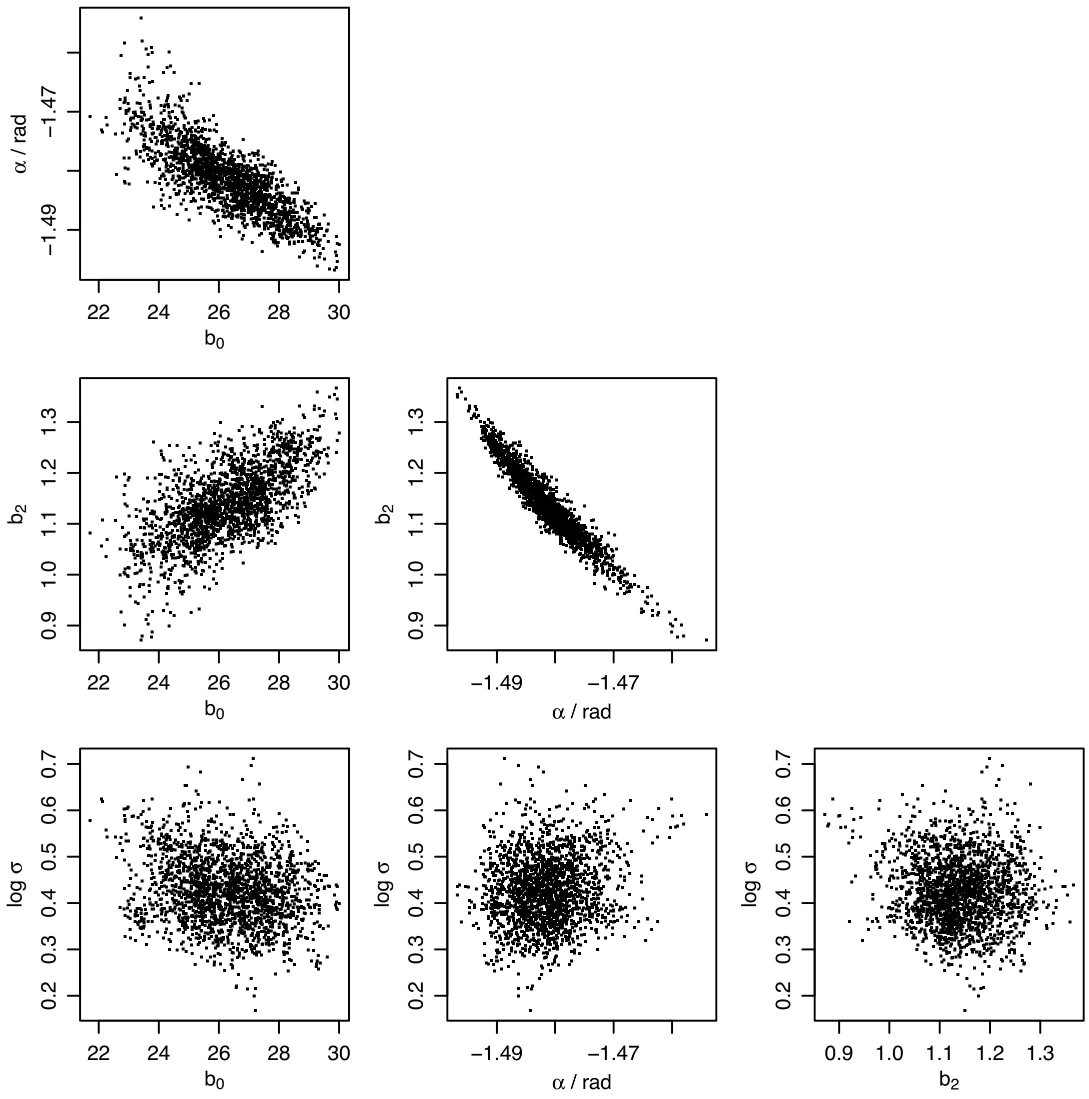

Fig. 9.12 The MCMC samples from the posterior PDF of the quadratic model, plotted in two dimensions to show the correlations between them.

as a *mixture model*

$$L_{\text{total}}(y) = (1-p)L_{\text{main}}(y) + pL_{\text{out}}(y) \quad \text{where} \quad 0 \le p \le 1 \tag{9.12}$$

where p is the *mixing factor*. The log likelihood for all data points is $\sum_i \log L_{\text{total}}(y_i)$. Our mixture model in principle has five parameters: b_0, α, σ, w, and p. With all other parameters fixed, we might think that a larger w would allow a better fit to outliers. But remember that the outlier distribution (equation 9.10) is normalized, so making it broader actually decreases the probability density at smaller values of $|y - f(x)|$. For a fixed w, the degree to which points are outliers can be controlled by the parameter p (to some extent – this will be discussed further below). I therefore fix $w = 1$ in the following.

We can now proceed with fitting a straight line as before (section 9.1), but with a different likelihood and an extra parameter, p. This needs a prior. As it is constrained to lie between 0 and 1, a natural choice is the beta distribution (section 1.4.3). Clearly a uniform prior is not appropriate: isn't the very nature of outliers that they are rare? In the code (see below) I set the two shape parameters of the beta distribution to be 1 and 20, which gives a strong peak at $p = 0$ and decreases monotonically to zero at $p = 1$ (overplotted later in figure 9.13 with the posterior). In a real application we should adjust this prior according to what we know about the frequency of outliers, and also investigate how sensitive the results are to the choice.

Having found the posterior PDFs, and having chosen a suitable estimator for the parameters, we can evaluate the probability that a particular point is an outlier by taking the ratio of the (unnormalized) probability that the point is an outlier to the (unnormalized) probability that the point is either an outlier or not

$$P_{\text{outlier}}(y) = \frac{pL_{\text{out}}(y)}{L_{\text{total(y)}}} = \frac{pL_{\text{out}}(y)}{(1-p)L_{\text{main}}(y) + pL_{\text{out}}(y)}. \tag{9.13}$$

The R code for fitting this model, `linearmodel_outlier_posterior.R`, is available online. It is similar in structure to that used for fitting the line without outliers (section 9.1.4), except that now I define the posterior function at the beginning of the file. The new parameter is the mixing factor, p. It must lie within the range 0–1, yet the MCMC sampler – which uses a multivariate Gaussian proposal distribution – can propose values outside of this range. These are automatically assigned zero prior probability by the `dbeta` function, but we must manually set L_{total} to zero (equation 9.12). The posterior is then zero, so this proposal will always be rejected by the Metropolis algorithm.

9.3.1 Application and results

I apply the model to the same data as that used in section 9.1, but with one point replaced with a single outlier. The outlier is ad hoc; it is not drawn from the outlier model. The MCMC is run for 10^5 iterations (with an initial burn-in of 10^4) and the resulting chains are thinned by a factor of 50. The resulting chains and one-dimensional marginalized posteriors are shown in figure 9.13. The chains are not particularly good (I have made little effort to optimize them), but they suffice for this demonstration. Figure 9.14 shows the original data (open circles) as well as the model fit from the maximum of the posterior (the solid line as well as the error bars). For comparison, the dashed line shows the least squares fit to all the data, i.e. without using an outlier model. We see that this is biased upwards by the presence of the outlier. Figure 9.15 shows the posterior samples in two dimensions. The only substantial correlation is between b_0 and α, for the same reason as when we had no outlier (the data are not centred).

The outlier probabilities for the points are shown in table 9.2. We see that not only does the mixture model prevent the outlier from distorting our fit too much, it also identifies the

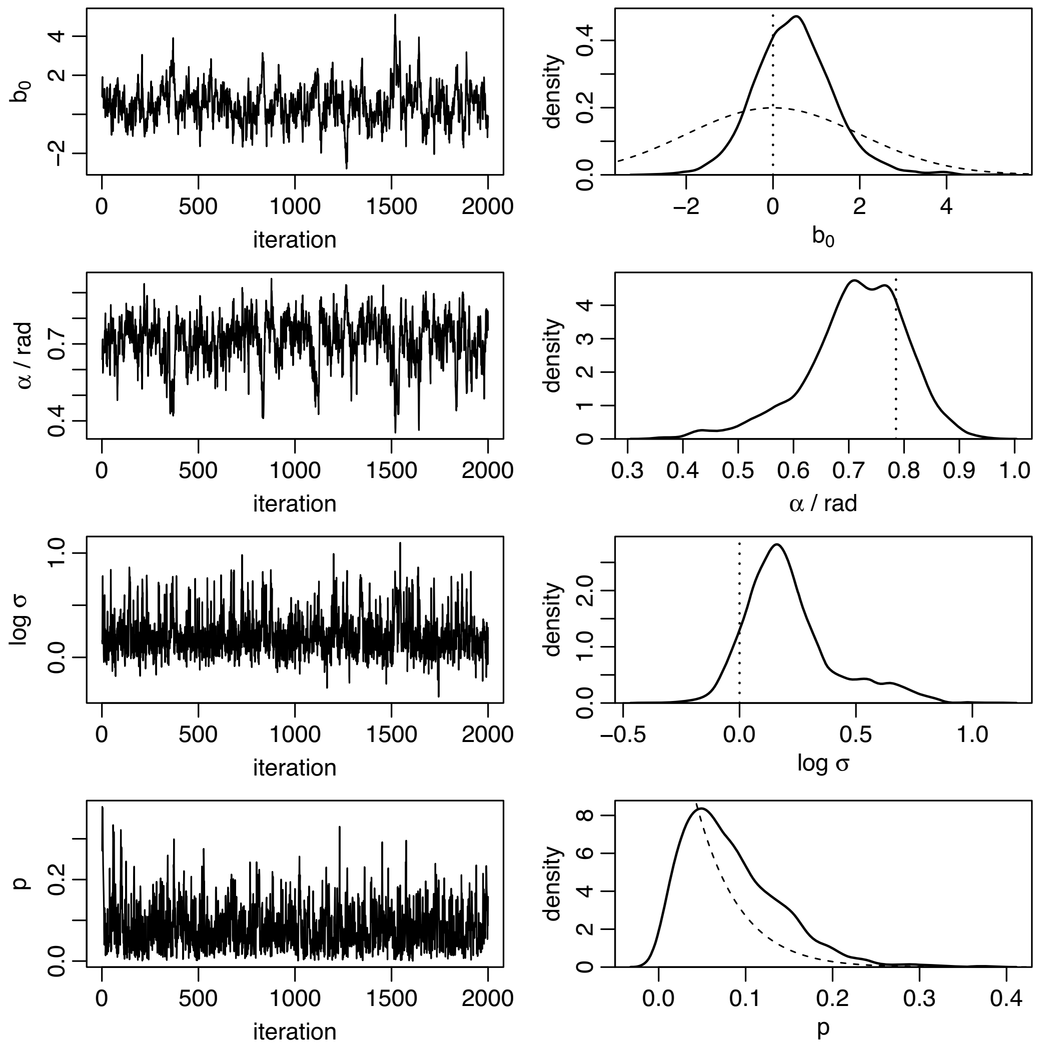

Fig. 9.13 MCMC chains (left columns) and resulting one-dimensional marginal posterior PDFs (right columns) for the straight line model with outlier model fitting problem. The one-dimensional posteriors have been computed by a kernel density estimate of the set of samples. The vertical dotted lines indicate the true parameters (there is no definition of the true value for the mixing coefficient p). The curved dashed lines in the panels for b_0 and p show the prior distributions for those parameters. The priors for the other two parameters are uniform.

outlier ($n = 5$). In this case identification would have been easy by eye, but this would be harder with a higher dimensional data set, a nonlinear problem, or for more marginal cases.

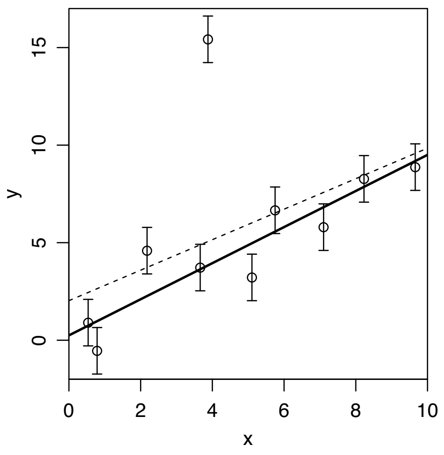

Fig. 9.14 Straight line fitting with an outlier model. The open circles are the data. These have been drawn from a straight line at fixed x values with $b_0 = 0$ and $b_1 = 1$ (equation 9.1). Zero mean Gaussian noise with standard deviation 1 has been added to all of these points, except the point at $x = 3.88$, which was made into an outlier by adding 10 to the function evaluation. The solid straight line and error bars show the model obtained with the parameters set to the maximum of the posterior PDF (MAP), which has values $b_0 = 0.25$, $b_1 = 0.93$, $\sigma = 1.19$, and $p = 0.048$. The dashed line is the least squares (maximum likelihood) solution to all the data, which assumes that all the data come from a Gaussian likelihood with common standard deviation.

9.3.2 Discussion

As outliers are presumably rare, the mixing factor p must be small. We might want to think of it as the average probability that a given point is an outlier *under this model*. Here I have fixed the value of the width parameter in the Cauchy distribution to 1. This is not robust, as the distribution is not scale independent: if we rescaled the data (e.g. measured y in different units), then we would need to change this parameter. However, provided it is on roughly the right scale, p will effectively act as the width parameter, because it scales the whole Cauchy distribution up and down, thus changing the contribution to the total likelihood. This has its limits, though, because p must be less than 1.

I do not claim that this outlier model is particularly good. I have used this simple method primarily to demonstrate the concept of using a likelihood mixture model.

There are several ways one could change or extend this approach. One is to make w a parameter that we infer along with the other four parameters. However, there is likely to be a correlation ("degeneracy") between w and p, as they can play a similar role. This is not a fundamental problem, but it might require many more MCMC samples to get good chains and suitable posterior PDFs. Another improvement would be to assign a different p to each data point (with w fixed again). This would not only give the model more flexibility, it

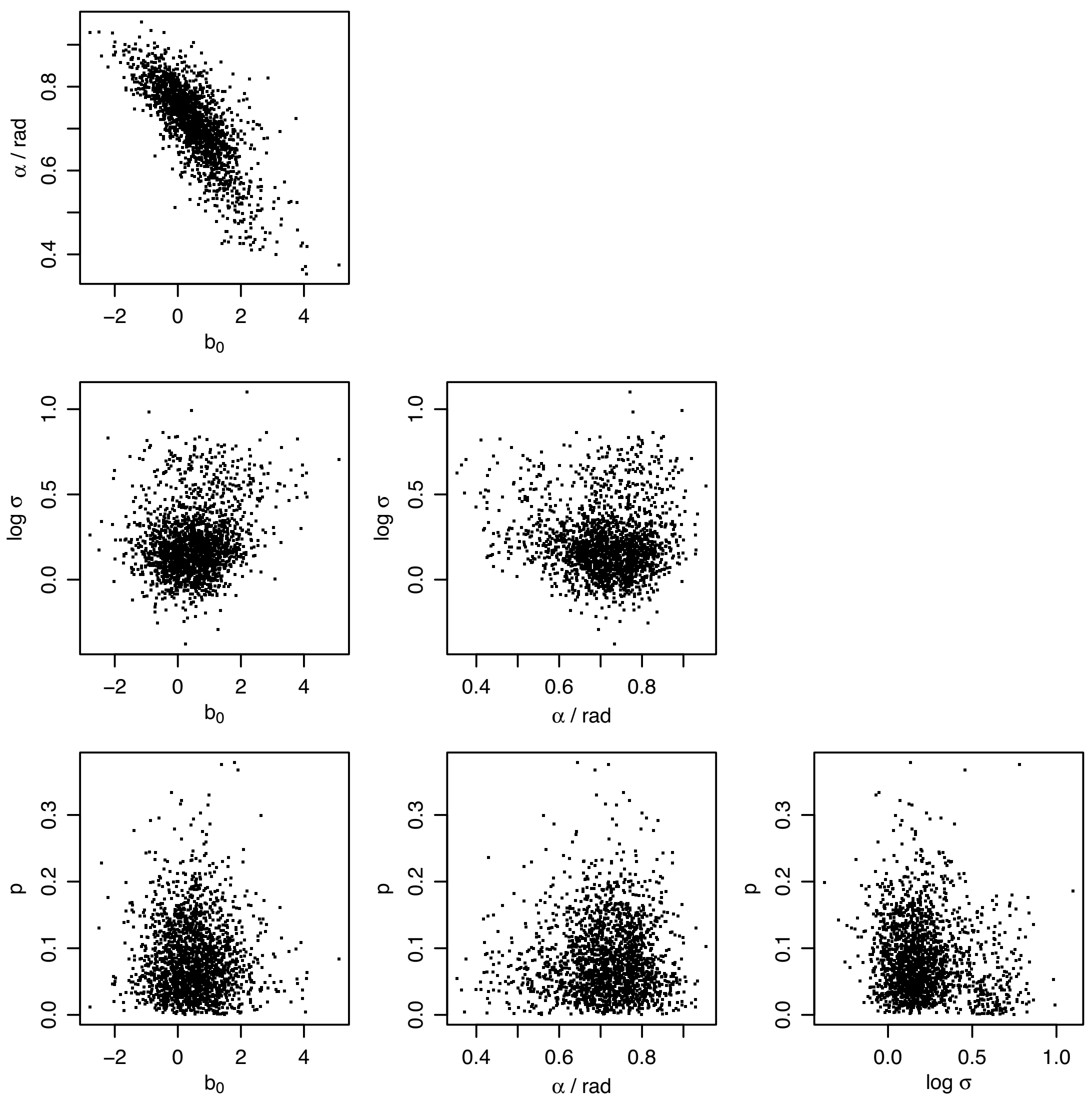

Fig. 9.15 The MCMC samples from the posterior PDF of the straight line model with an outlier model, plotted in two dimensions to show the correlations between them.

would allow us (but not force us) to assign a different prior to each point, which is useful if we are more suspicious of some points that others. This would increase the number of parameters, and in fact increase it to be more than the number of data points. This is not a problem for a Bayesian analysis per se. But such flexibility can present new difficulties, not least for the MCMC because it now has to sample a much higher dimensional space. This often requires many more samples and better tuning of the step sizes. (Metropolis is not well suited to sampling parameters spaces with many dimensions.)

Table 9.2 Outlier probabilities (equation 9.13) for the ten points shown in figure 9.14

i	x	y	P_{outlier}
1	0.5	0.9	0.045
2	0.8	−0.5	0.032
3	2.2	4.6	0.048
4	3.7	3.7	0.045
5	3.9	15.4	1.000
6	5.1	3.2	0.033
7	5.8	6.7	0.032
8	7.1	5.8	0.033
9	8.2	8.3	0.042
10	9.7	8.9	0.043

9.4 Fitting curves with arbitrary error bars on both axes

Sometimes we have uncertainties not only in the y values but also in the x values. We saw one way to deal with this in a least squares sense in section 4.7. The Bayesian approach is more general and more flexible. For example, it allows us to accommodate any noise model, not just a Gaussian one.

A general approach is to make a distinction between the (noisy) measured values of x and y and their true (noise-free) but unknown values, which I will denote x' and y' respectively. Thus

$$\begin{aligned} x &= x' + \epsilon_x \\ y &= y' + \epsilon_y \end{aligned} \tag{9.14}$$

where ϵ_x and ϵ_y are random numbers which in general depend on both x' and y'. We then write the noise model, with parameters ϕ, for a single data point as $P(x, y \,|\, x', y', \phi)$. An example of this is a bivariate Gaussian distribution, perhaps with correlated errors in x and y, in which case ϕ is the covariance matrix. As before the generative model relates the noise-free quantities as $y' = f[x'; \theta]$, where θ are its parameters. This could be a polynomial model or a nonlinear model, for example. Whatever the noise and generative models, we can write the likelihood for a single data point as a marginalization over the true, unknown values

$$P(x, y \,|\, \theta, \phi) = \iint P(x, y \,|\, x', y', \phi)\, P(x', y' \,|\, \theta)\, dx' dy'. \tag{9.15}$$

The second term under the integral we can write as

$$P(x', y' \,|\, \theta) = P(y' \,|\, x', \theta) P(x' \,|\, \theta). \tag{9.16}$$

In many situations we may assume that, prior to making any measurements, all values of x' are equally probable over an arbitrarily large range of x' between $x'_{\rm lo}$ and $x'_{\rm hi}$. In that case we can set $P(x'\,|\,\theta) = 1/\Delta x'$ inside this range, and zero outside the range, where $\Delta x' = x'_{\rm hi} - x'_{\rm lo}$. We will see in a moment that none of these values actually matter. If we further assume that the generative model is deterministic (we have only considered such models in this book), then

$$P(y'\,|\,x', \theta) \;=\; \delta(y' - f[x'; \theta]) \tag{9.17}$$

where $\delta()$ is the delta function. Inserting this into equation 9.16 and that into equation 9.15, we see that the integration over y' is trivial on account of the delta function: the integral is only non-zero at $y' = f[x'; \theta]$. Thus the likelihood for one data point is

$$P(x, y\,|\,\theta, \phi) \;=\; \frac{1}{\Delta x'} \int_{x'_{\rm lo}}^{x'_{\rm hi}} P(x, y\,|\,x', y' \!=\! f[x'; \theta], \phi)\; dx'. \tag{9.18}$$

This likelihood is normalized,[7] so it must become arbitrarily small at extreme values of x. Thus provided $\Delta x'$ is large enough – and we can make it arbitrarily large – then the values of $x'_{\rm lo}$ and $x'_{\rm hi}$ are irrelevant, because the integrand in equation 9.18 will become arbitrarily small before we hit these limits. Furthermore, $\Delta x'$ will often be independent of θ, in which case it can be absorbed into the normalization constant for the posterior. We just have to ensure that the integration over x' extends out to where the likelihood becomes very small.

If we have a set of N data points $D = \{x_i, y_i\}$ that have been measured independently, then the likelihood of all the data is

$$P(D\,|\,\theta, \phi) \;=\; \prod_i P(x_i, y_i\,|\,\theta, \phi). \tag{9.19}$$

The unnormalized posterior is therefore

$$P^*(\theta, \phi\,|\,D) \;=\; P(\theta, \phi) \prod_i \int_{x'_{\rm lo}}^{x'_{\rm hi}} P(x_i, y_i\,|\,x', y' \!=\! f[x'; \theta], \phi)\; dx' \tag{9.20}$$

where $P(\theta, \phi)$ is the prior.

We can now proceed as in the earlier sections: we decide on a prior then sample from the posterior using MCMC. The only difference now is that each evaluation of the posterior involves N one-dimensional integrations. In general these will need to be solved numerically, so the posterior may be computationally expensive (slow) to compute. Yet this approach allows us to fit arbitrary curves to data with arbitrary noise models on both x and y.

To illustrate this method, let us suppose we want to fit a straight line

$$y' \;=\; f[x'] \;=\; b_0 + b_1 x' \tag{9.21}$$

to some data $D = \{x, y\}$, for which we have known Gaussian uncertainties in x and y of σ_x and σ_y respectively. These are the same for each data point. Thus the noise model

[7] This is important, because in general the normalization constant of the likelihood depends on the parameters θ, and so cannot be ignored when we are interested in the posterior $P(\theta\,|\,D)$.

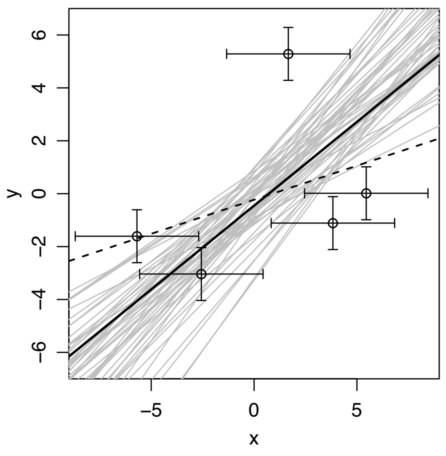

Fig. 9.16 Straight line fitting with uncertainties in both x and y. The open circles are the data, and the error bars show the (known) one-sigma uncertainties. The solid black line shows the straight line model corresponding to the maximum of the posterior. The grey lines show 50 models drawn from the posterior. The dashed line is the least squares fit ignoring the uncertainties in x.

parameters ϕ are fixed. The likelihood for one data point is therefore given by a bivariate Gaussian

$$P(x, y \,|\, x', y' = f[x'; \theta]) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[(x,y) - (x',y')]^{\mathsf{T}}\Sigma^{-1}[(x,y) - (x',y')]\right)$$

where

$$\Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \tag{9.22}$$

We could easily generalize this approach to have different known uncertainties in x and y for each point, and even a different non-zero correlation for each data point. Alternatively, we could let σ_x, σ_y, and the correlation between them be unknown parameters that we want to determine from the data. But for now we assume there is just a single σ_x and a single σ_y, and these are known. The parameters of this model are therefore (b_0, b_1).

I simulate some data by drawing five points from a uniform distribution between 0 and 10. These are the noise-free x values. The corresponding noise-free y values are computed from equation 9.21 with $b_0 = 0$ and $b_1 = 1$. I then add zero mean Gaussian noise with standard deviation $\sigma_x = 3$ to the noise-free x values to get the x data (and then I centre these; see section 9.1.6). I likewise add zero mean Gaussian noise with standard deviation $\sigma_y = 1$ to noise-free y values to get the y data (and also centre these). I do not use the noise-free data in the inference, of course.

The simulated data are shown in figure 9.16. I adopt a uniform prior on $\alpha = \arctan b_1$, and a Gaussian prior on b_0 with mean zero and standard deviation 5. I use the Metropolis algorithm to sample the posterior over the parameters (b_0, α), using a Gaussian proposal

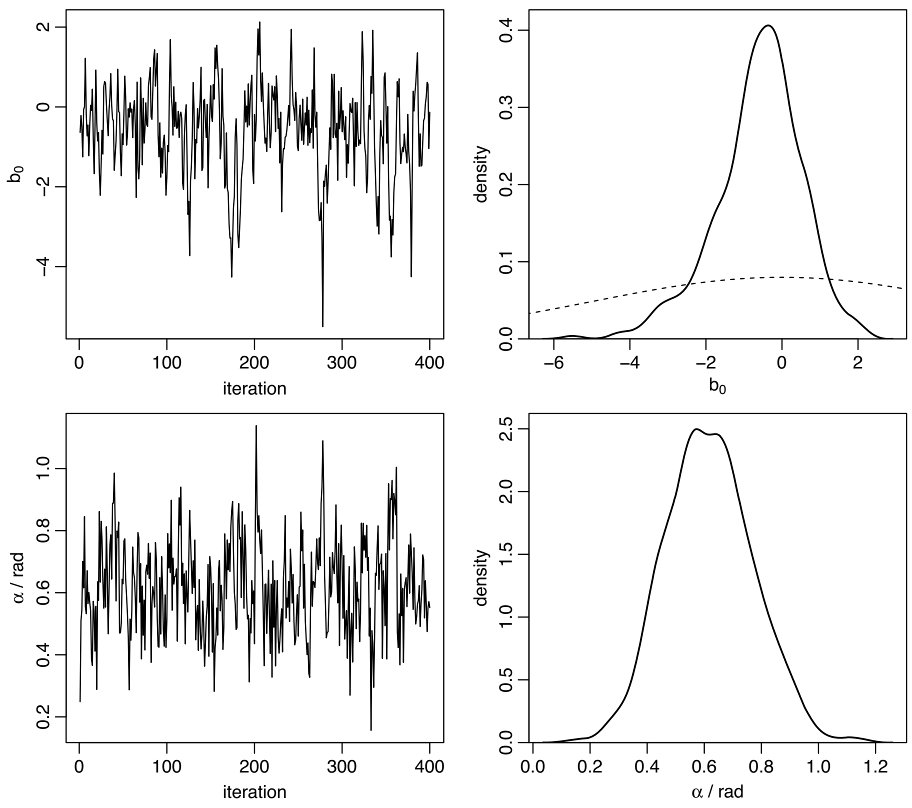

Fig. 9.17 MCMC chains (left columns) and resulting marginal posterior PDFs (right columns) for fitting a straight line with uncertainties in both x and y. These one-dimensional posteriors have been computed by a kernel density estimate of the samples. The curved dashed line in the panel for b_0 shows the prior distribution. The prior over α is uniform.

distribution with standard deviations (step sizes) of $(0.25, 0.05)$. I initialize Metropolis at the least squares solution for b_0 and α, sample for 10 000 iterations, and apply a thinning factor of 25. (Each iteration involves five numerical integrations.) This took about ten minutes to run on my laptop.

The resulting chains and marginal posteriors are shown in figure 9.17.[8] The posteriors are quite broad, i.e. the parameters are not precisely determined, but this is not surprising given that we have only five data points and large error bars. The straight line corresponding to the maximum of the posterior is shown as the thick solid black line in figure 9.16. The thin grey lines show 50 models drawn from the posterior to illustrate how the uncertainty represented by the finite width of the posterior transforms to the data space. The dashed

[8] Although the chains have been thinned, they still show some signs of correlations, even though the correlation lengths (post-thinning) are about 5 and 2 for b_0 and α respectively.

line shows the ordinary least squares solution (section 4.1). It is quite different, primarily because it ignores the uncertainties in x.

The online R file `linearmodel_xyerr_posterior.R` performs the above experiment. This also makes use of the file `linearmodel_xyerr_functions.R`. These are analogous to the two files of similar name listed in section 9.1.4. The one-dimensional integral in equation 9.18 is done numerically using the R function `integrate` (which uses the technique of adaptive quadrature). We can think of this as integrating a bivariate Gaussian along the line $y' = f[x'; \theta]$.

The approach described above up to and including equation 9.16 is completely general. I then assumed a uniform prior for $P(x' \mid \theta)$. This could of course be generalized. For example, if the data were a time series, such that x is time and y is the signal, then the phenomenon being measured might be one in which events are more likely to occur at some times than others, or maybe cannot occur in certain time ranges at all. The phenomenon itself might even be non-deterministic, in which case we could use a stochastic model for $f[x'; \theta]$. The principle of marginalizing over x' and y' to determine the likelihood for each data point (x, y) still applies. For those who want to know more, see Bailer-Jones (2012).