Chapter 13

Random Walk Metropolis

13.1 Ticked off

Imagine once again that you are investigating the occurrence of Lyme disease in the UK. This is a vector-borne disease caused by bacteria of species *Borrelia* which is carried by ticks. (The ticks pick up the infection by blood-feeding on animals or humans that are infected with *Borrelia*.) You decide to estimate the prevalence of this bacteria in ticks you collect from the grasslands and woodlands around Oxford.

You decide to use sample sizes of 100 ticks, out of which you count the number of ticks testing positive for *Borrelia*. You decide to use a binomial likelihood since you assume that the presence of *Borrelia* in one tick is independent of that in other ticks. Also because you sample a relatively small area you assume that the presence of *Borrelia* can be assumed to be identically-distributed across ticks.

Problem 13.1.1. You specify a $beta(1,1)$ distribution as a prior. Use independent sampling to estimate the prior predictive distribution (the same as the posterior predictive except using sampling from the prior in the first step rather than the posterior), and show that its mean is approximately 50.

This distribution is a discrete uniform distribution between 0 and 100 ticks. To estimate this distribution we first of all draw a value of $\theta_i \sim beta(1,1)$, then draw a random sample $X_i \sim \mathcal{B}(100, \theta_i)$. Repeating this exercise a few thousand times we get a reasonably accurate prior predictive distribution. To do this in R,

```
fPriorPredictive <- function(numSamples, a, b, N){
  lX <- vector(length=numSamples)
  for(i in 1:numSamples){
    theta <- rbeta(1, a, b)
    lX[i] <- rbinom(1, N, theta)
  }
  return(lX)
}
```

```
lX <- fPriorPredictive(1000, 1, 1, 100)
hist(lX)
mean(lX)
```

Problem 13.1.2. In a single sample you find that there are 6 ticks that test positive for *Borrelia*. Assuming a $beta(1,1)$ prior graph the posterior distribution, and find its mean.

Since the beta prior here is conjugate to the binomial likelihood the posterior is also a beta distribution. To transform a beta prior into a posterior, we use the rule: $beta(a,b) \rightarrow beta(a+X, b+n-X)$, where (a,b) are the prior parameters, X is the number of ticks collected and n is the sample size. Since we are using a beta(1,1) prior here, the posterior is a beta(7,95) distribution; which has a posterior mean of $\frac{7}{102} \approx 0.069$.

Problem 13.1.3. Generate 100 independent samples from this distribution using your software's inbuilt (pseudo-)random number generator. Graph this distribution. How does it compare to the PDF of the exact posterior? (Hint: in R the command is "rbeta"; in Matlab it is "betarnd"; in Mathematica it is "RandomVariate[BetaDistribution...]"; in Python it is "numpy.random.beta".)

After only 100 independent samples the estimated posterior is quite similar in shape to the actual distribution (Figure 13.1).

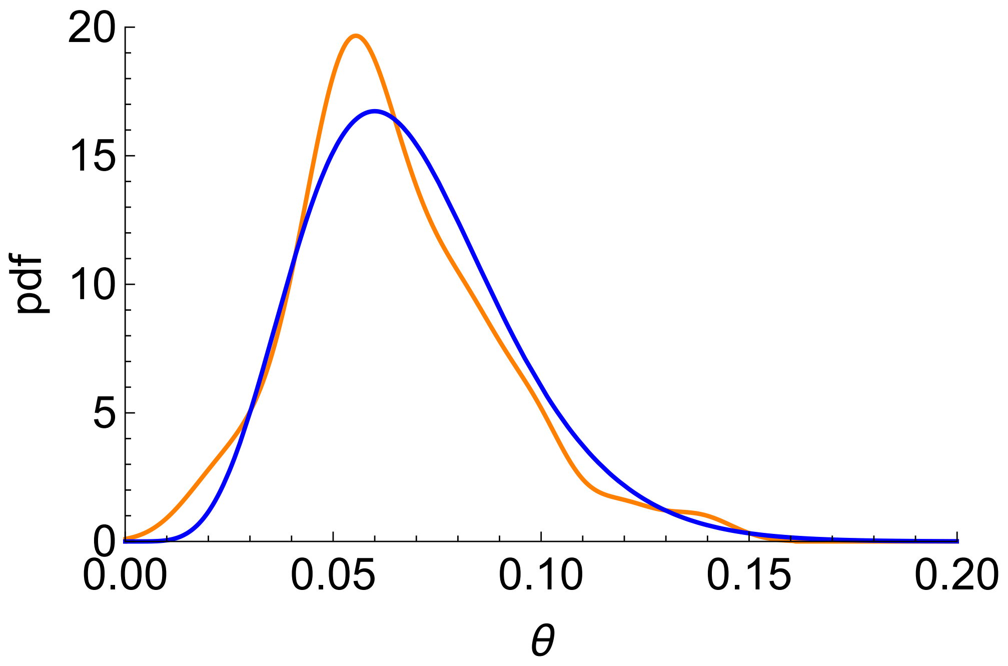

Figure 13.1: A PDF of the posterior estimated from independent samples (orange) versus the exact posterior (blue).

Problem 13.1.4. Determine the effect of increasing the sample size on using the independent sampler to estimate the posterior mean. (Hint: for each sample you are essentially comparing the sample mean with the true mean of the posterior.)

The error in using independent sampling to estimate the mean of a distribution is given by the (Lindberg-Lévy) central limit theorem:

$$(\bar{X} - \mathbb{E}[X]) \xrightarrow{d} \mathcal{N}(0, \sigma) \tag{13.1}$$

which means we can estimate the error for a large sample of size n by:

$$(\bar{X} - \mathbb{E}[X]) \approx \mathcal{N}(0, \frac{\sigma}{\sqrt{n}}) \tag{13.2}$$

This means that as the sample size increases the error in estimation decreases in accordance with $\sqrt{n}$ (Figure 13.2).

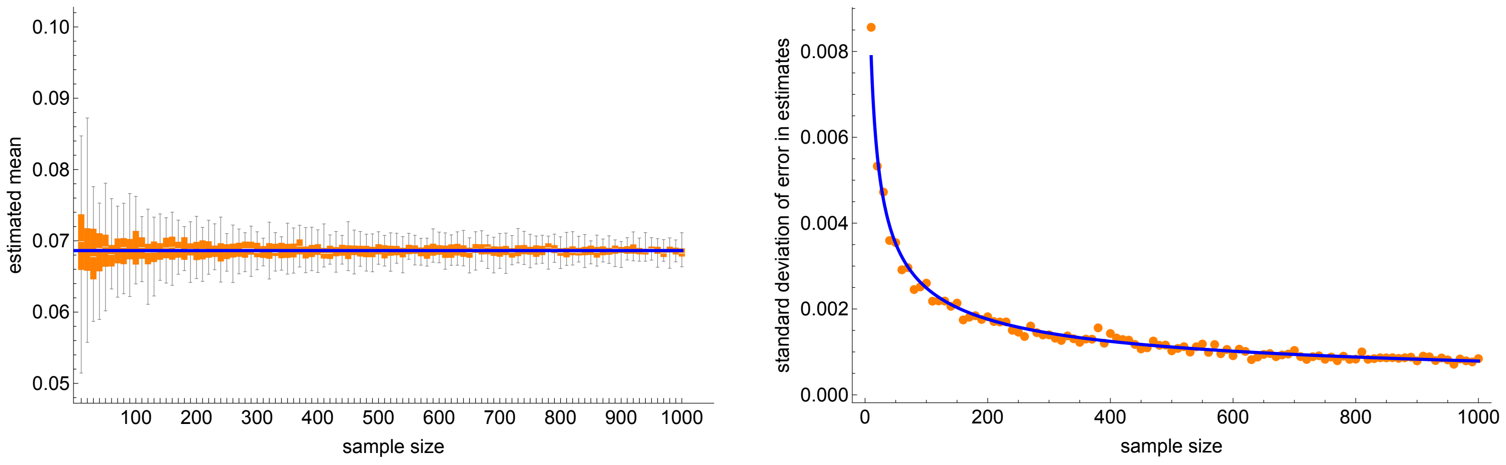

Figure 13.2: The estimated mean (left) and standard deviation in the error (right) using independent sampling to estimate the mean of the posterior for the ticks example.

Problem 13.1.5. Estimate the variance of the posterior using independent sampling for a sample size of 100. How does your sample estimate compare with the exact solution?

To do this generate an independent sample of size 100, and calculate the sample variance (see Figure 13.3.) Even after 100 samples we are able to estimate the posterior variance with quite reasonable resolution.

Problem 13.1.6. Create a proposal function for this problem that takes as input a current value of θ, along with a step size, and outputs a proposed value. For a proposal distribution here we use a normal distribution centred on the current θ value with a standard deviation (step size) of 0.1. This means you will need to generate a random θ from a normal distribution using your statistical software's inbuilt random number generator. (Hint: the only slight modification you need to make here is to ensure that we don't get $\theta < 0$ or $\theta > 1$ is to use periodic boundary conditions. To do this we use modular arithmetic. In particular we set $\theta_{proposed} = mod(\theta_{proposed}, 1)$. The command for this in R is $x\%\%1$; in Matlab the command is $mod(x, 1)$; in Mathematica it is $Mod[x, 1]$; in Python it is $x\%1$.)

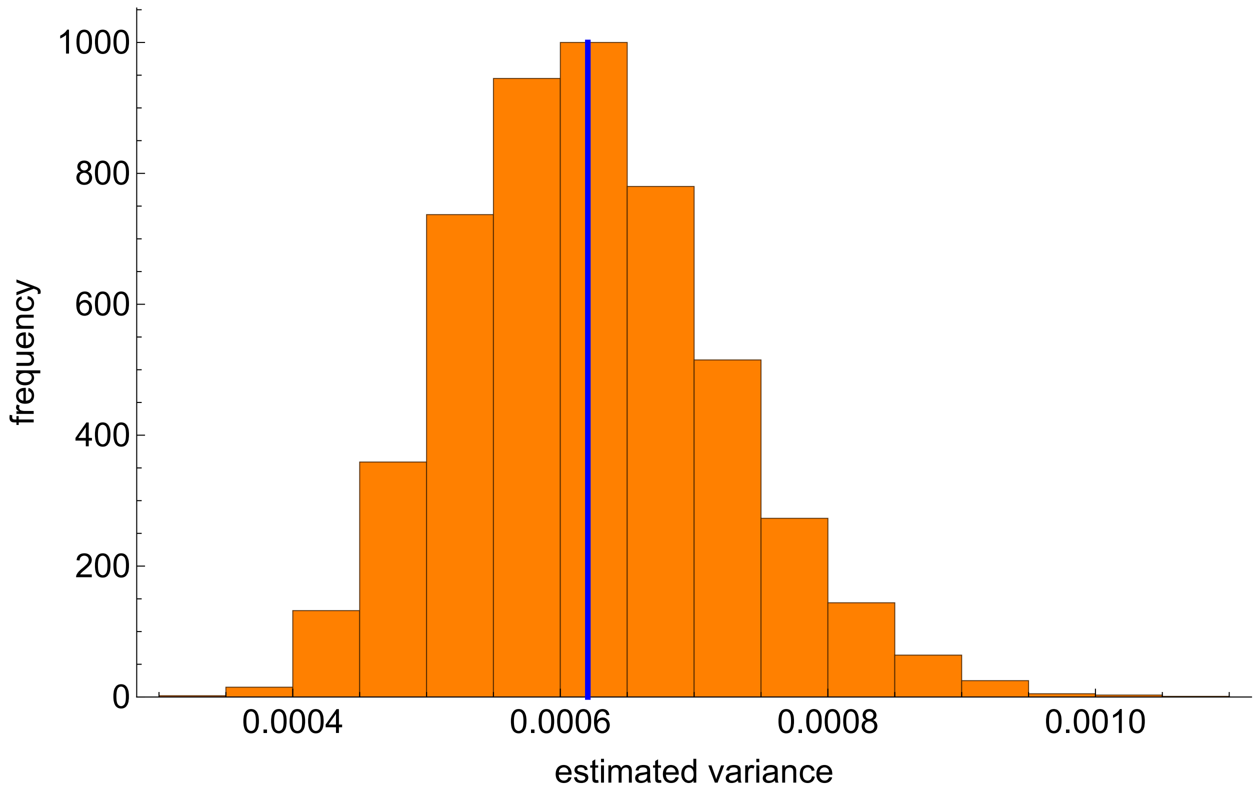

Figure 13.3: The estimated variance of the posterior for the ticks example versus the actual value (blue line) for an independent sample of size 100.

Problem 13.1.7. Create the "accept/reject" function of Random Walk Metropolis that accepts as input $\theta_{current}$ and $\theta_{proposed}$ and outputs the next value of θ. This is done based on a ratio:

$$r = \frac{p(X|\theta_{proposed}) \times p(\theta_{proposed})}{p(X|\theta_{current}) \times p(\theta_{current})} \tag{13.3}$$

and a uniformly-distributed random number between 0 and 1, which we call a. If $r > a$ then we update our current value of $\theta_{current} \rightarrow \theta_{proposed}$; alternatively we remain at $\theta_{current}$.

Problem 13.1.8. Create a function that combines the previous two functions; so it takes as input a current value of $\theta_{current}$, generates a proposed $\theta_{proposed}$, and updates $\theta_{current}$ in accordance with the Metropolis accept/reject rule.

Problem 13.1.9. Create a fully working Random Walk Metropolis sampler. (Hint: you will need to iterate the last function. Use a uniformly distributed random number between 0 and 1 as a starting point.)

Problem 13.1.10. For a sample size of 100 from your Metropolis sampler compare the sampling distribution to the exact posterior. How does the estimated posterior compare with that obtained via independent sampling using the same sample size?

The MCMC sample distribution is not as crisp as the independent sample distribution (Figure 13.4). This is because of the effects of dependence on the sampling efficiency. Intuitively, the information conveyed from each incremental sample is less than for the independent case.

There is also a slight bias in the MCMC posterior towards the starting point of the algorithm. This is because sampler hasn't had sufficient time to converge to the posterior. This bias can be removed by using more samples from the posterior and discarding those samples during the "warm-up" period.

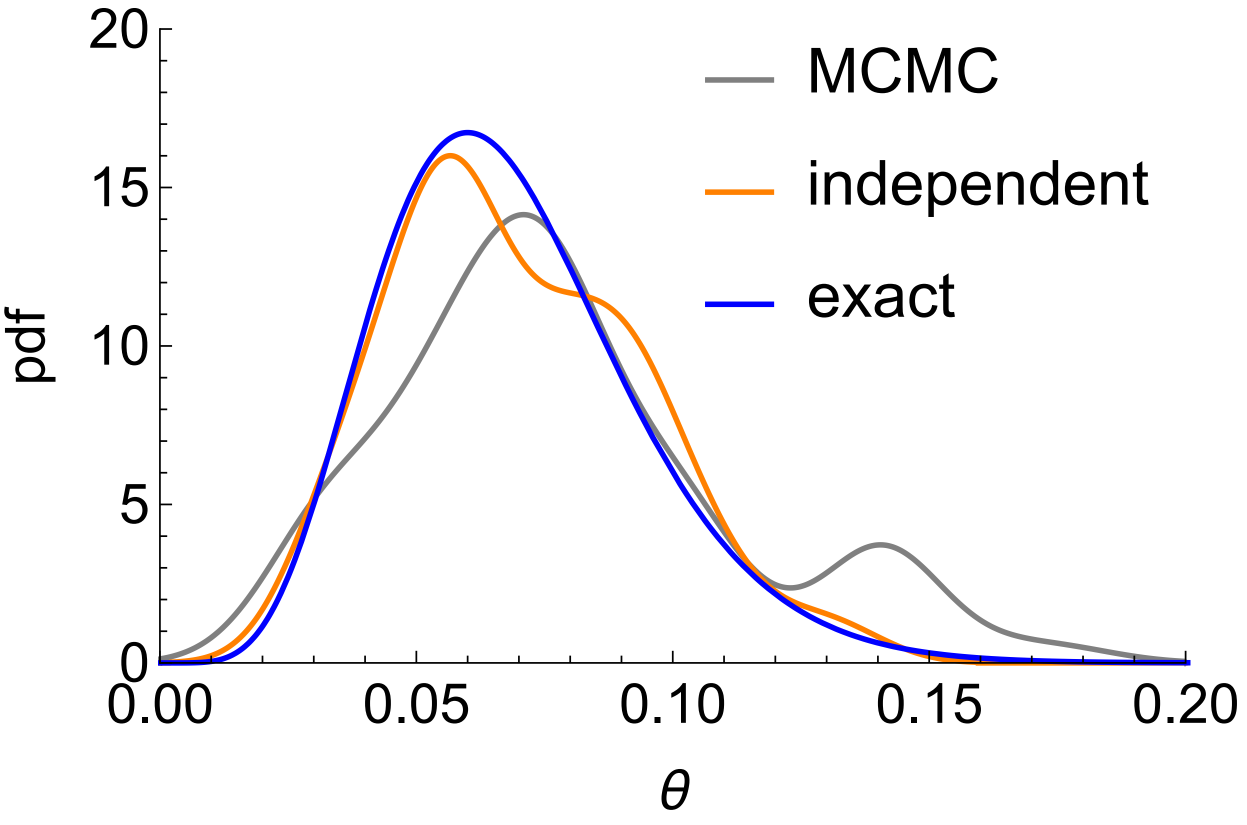

Figure 13.4: The estimated posterior via MCMC (orange) and independent (grey) sampling versus the exact posterior (blue).

Problem 13.1.11. Run 1000 iterations, where in each iteration you run a single chain for 100 iterations. Store the results in a 1000 x 100 matrix. For each iterate calculate the sample mean. Graph the resultant distribution of sample means. Determine the accuracy of the MCMC at estimating the posterior mean?

With only 100 samples we have not given the chains sufficient time to converge to the posterior density; specifically the effect of using a random start position that is *not* from the posterior means that our posteriors still reflect this. Therefore if we calculate the mean of our 100 posterior samples, it will tend to be upwardly biased of the true value because we haven't allowed for the "warm-up" period (Figure 13.5).

Problem 13.1.12. Graph the distribution of the sample means for the second 50 observations of each chain. How does this result compare with that of the previous question? Why is there a difference?

The difference is solely due to the warm-up period being discarded (Figure 13.5). We have allowed the chains time to converge to the posterior, and hence by discarding the first 50 observations we reduce the effect of the random starting position. Since we are now using converged chains the estimator of the mean is unbiased.

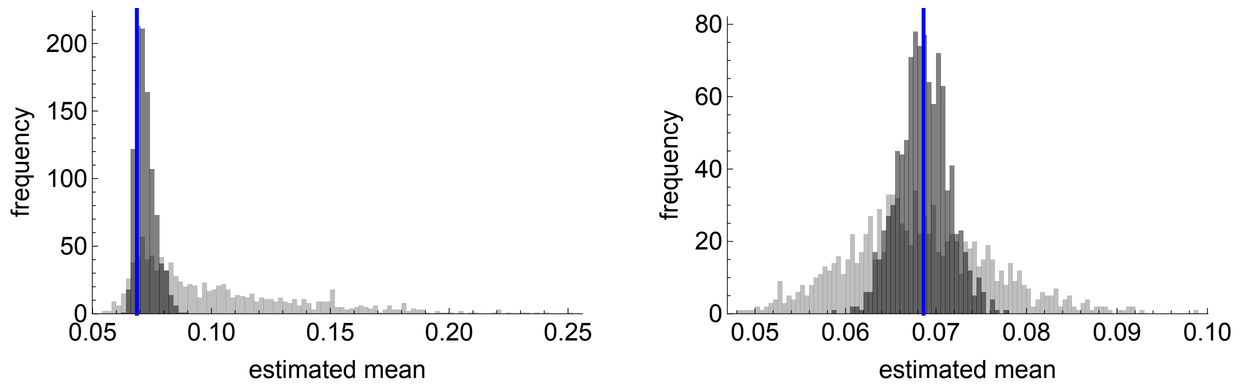

Figure 13.5: The sampling distribution of the sample mean of the MCMC runs for the ticks example, where (left) we use all 100 (light grey) or 1000 samples (dark grey) from each chain, and (right) we only use the second half of each.

Problem 13.1.13. Decrease the standard deviation (step size) of the proposal distribution to 0.01. For a sample size of 200, how the posterior for a step size of 0.01 compare to that obtained for 0.1?

A step size of 0.1 is able to find, then explore, the typical set at a much faster rate than the smaller step size (Figure 13.6); meaning that there is a lot of autocorrelation in the sampler's value. Intuitively, a sampler with a small step size is not able to move far from where it was at the end of the previous iteration!

Basically using a step size that is too low is equivalent to using a toothbrush in an archaeological dig. It takes you ages to find any hidden treasures!

Problem 13.1.14. Increase the standard deviation (step size) of the proposal distribution to 1. For a sample size of 200, how does the posterior for a step size of 1 compare to that obtained for 0.1?

Now the sampler is able to find the typical set fast enough. The trouble now is that it is inefficient at exploring it (Figure 13.7). Intuitively, the path of the sampler is characterised by a high rejection rate, since most of the proposed steps are a long way away from the region of high density.

Overall this means that the reconstructed probability mass at least lies in the correct region of parameter space. However, the reconstructed density has a high variance because there are relatively few unique samples relative to the density from a step size of 0.1.

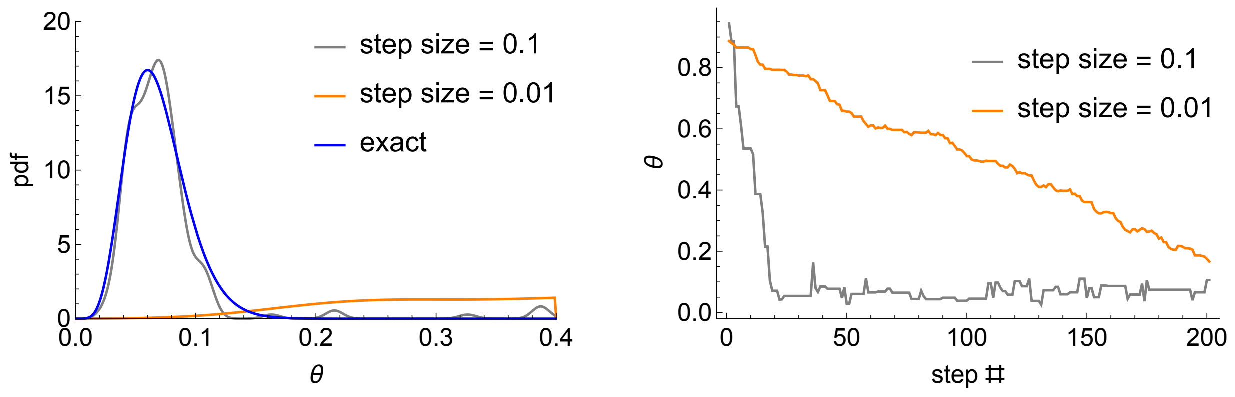

Figure 13.6: Left: the estimated posteriors for MCMC runs using two step sizes versus the actual. Right: the evolution of the path of each Markov Chain over time.

Basically using a step size that is too large is equivalent to using a digger in an archaeological dig; it finds the treasure fast enough but is too crude to save its finer details.

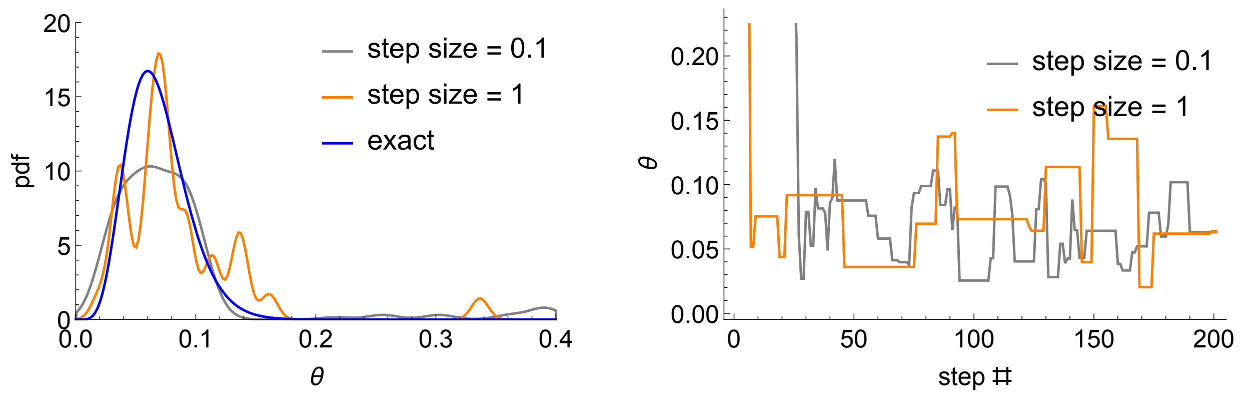

Figure 13.7: Left: the estimated posteriors for MCMC runs using two step sizes versus the actual. Right: the evolution of the path of each Markov Chain over time.

Problem 13.1.15. Suppose we collect data for a number of such samples (each of size 100), and find the following numbers of ticks that test positive for *Borrelia*: (3,2,8,25). Either calculate the new posterior exactly, or use sampling to estimate it. (Hint: in both cases make sure you include the original sample of 6.)

The posterior is a $beta(45, 457)$ distribution, which has a mean at about $\theta = 0.09$.

Problem 13.1.16. Generate samples from the posterior predictive distribution, and use these to test your model. What do these suggest about your model's assumptions?

Posterior predictive samples are unable to well-replicate either the minimum nor the maximum in the data (Figure 13.8). These suggest that either the assumption of independence or identical-distribution is violated; both of which there are good reasons for! (If one tick has the bacteria it will infect nearby animals and in doing so, make it more likely for other ticks to become infected; meaning independence is likely violated. Following on from this we probably think that due to the contagious nature of the disease that there will be hotspots; meaning that the assumption of identical distribution is likely violated.)

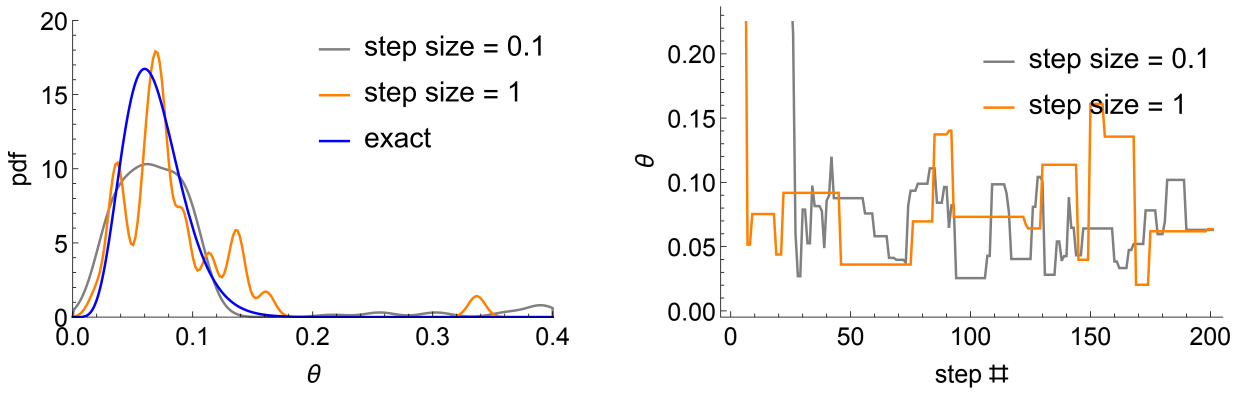

Figure 13.8: The posterior predictive distribution for a dataset of (6,3,2,8,25) *Borrelia*-positive ticks; each out of a sample of 100.

Problem 13.1.17. A colleague suggests as an alternative you use a beta-binomial likelihood, instead of the existent binomial likelihood. This distribution has two uncertain parameters $\alpha > 0$ and $\beta > 0$ (the other parameter is the sample size; $n = 100$ in this case), where the mean of the distribution is $\frac{n\alpha}{\alpha+\beta}$ Your colleague and you decide to use weakly informative priors of the form: $\alpha \sim \Gamma(1, \frac{1}{8})$ and $\beta \sim \Gamma(10, 1)$. (Here we use the parameterisation such that the mean of $\Gamma(a, b) = \frac{a}{b}$.) Visualise the joint prior in this case.

The joint prior is just the product of the individual priors because of independence between the parameters.

Problem 13.1.8. For this situation your colleague tells you that there are unfortunately no conjugate priors. As such, three possible solutions (of many) open to you are: 1. you use numerical integration to find the posterior parameters, or 2. use the Random Walk Metropolis-Hastings algorithm, or 3. you transform each of (α, β) so that they lie between $-\infty < \theta < \infty$. Why can't you use vanilla Random Walk Metropolis for (α, β) here?

The trouble is that the parameters are bounded to only be non-negative. Previously our parameter θ was bounded between 0 and 1, but we got around the issue of its bounds by using periodic boundary conditions; preserving the symmetry of the proposal distribution. This meant the we could just use vanilla Random Walk Metropolis with good sampling efficiency.

Here the problem is it is not possible to use periodic boundary conditions because there is only one boundary. This means that if we sample from (α, β) directly then we would ideally use an asymmetric proposal distribution (for example the log-normal); meaning we use the Metropolis-Hastings algorithm. Alternatively we transform (α, β) so that the transformed parameters are unbounded. Finally, because the model is fairly simple, having only two parameters, we can actually use numerical integration here to find the posterior.

Problem 13.1.19. By using one of the three methods above estimate the joint posterior distribution. Visualise the PDF of the joint posterior. How are α and β correlated here?

I used numeric integration here because the model is simple enough for it (see Figure 13.9). The Mathematica code is shown below,

```
fPrior1[\[Alpha]_, \[Beta]_] :=
  PDF[GammaDistribution[1, 8], \[Alpha]] PDF[
  GammaDistribution[8, 1], \[Beta]]

fLikelihood1[\[Alpha]_, \[Beta]_, x__, n_Integer] :=
  Likelihood[BetaBinomialDistribution[\[Alpha], \[Beta], n], x]

fPosteriorUnnormalised1[\[Alpha]_, \[Beta]_, x__, n_Integer] :=
  fPrior1[\[Alpha], \[Beta]] fLikelihood1[\[Alpha], \[Beta], x, n]

aInt = NIntegrate[
  fPosteriorUnnormalised1[\[Alpha], \[Beta], newdata,
    100], {\[Alpha], 0, \[Infinity]}, {\[Beta], 0, \[Infinity]}];

fPosterior1[\[Alpha]_, \[Beta]_, x__, n_Integer, aInt_] :=
  fPosteriorUnnormalised1[\[Alpha], \[Beta], x, n]/aInt

g1 = ContourPlot[
  fPosterior1[\[Alpha], \[Beta], newdata, 100, aInt], {\[Alpha], 0,
    2.1}, {\[Beta], 0, 20}, PlotRange -> Full, Evaluate@options3,
    FrameLabel -> {"\[Alpha]", "\[Beta]"}]
```

The parameters are positively correlated in Figure 13.9. This makes sense because the mean of the beta-binomial is $\frac{n\alpha}{\alpha+\beta}$. So if we increase α we need to increase β to ensure the mean is maintained, and we still are able to fit the data.

I have also used the other two methods; I used a log-normal jumping distribution for the Metropolis-Hastings set up. Here we use a log-normal whose mean is the current position of the Markov Chain. This is non-trivial because the mean of a log-normal isn't μ. The Mathematica code to do this is,

```
fStepMH[\[Alpha]_, \[Beta]_, aStepSize_] :=
  RandomVariate[
    LogNormalDistribution[1/2 (-aStepSize^2 + 2 Log[#]),
    aStepSize], {1}][[1]] & /@ {\[Alpha], \[Beta]}

fAcceptMH[lCurrent__, lProposed__, x__, n_Integer, aStepSize_] :=
  Module[{r = (
    fPosteriorUnnormalised1[lProposed[[1]], lProposed[[2]], x, n]/
    fPosteriorUnnormalised1[lCurrent[[1]], lCurrent[[2]], x, n])
    (PDF[LogNormalDistribution[
    1/2 (-aStepSize^2 + 2 Log[lProposed[[1]]]), aStepSize],
    lCurrent[[1]]] PDF[
    LogNormalDistribution[
    1/2 (-aStepSize^2 + 2 Log[lProposed[[2]]]), aStepSize],
    lCurrent[[2]]])/(PDF[
    LogNormalDistribution[
    1/2 (-aStepSize^2 + 2 Log[lCurrent[[1]]]), aStepSize],
    lProposed[[1]]] PDF[
    LogNormalDistribution[
    1/2 (-aStepSize^2 + 2 Log[lCurrent[[2]]]), aStepSize],
    lProposed[[2]]]), aRand = RandomReal[]},
    If[r > aRand, lProposed, lCurrent]]

fTakeStepMH[lCurrent__, x__, n_Integer, aStepSize_] :=
  Module[{lProposed =
    fStepMH[lCurrent[[1]], lCurrent[[2]], aStepSize]},
    fAcceptMH[lCurrent, lProposed, x, n, aStepSize]]

fMetropolisHastings[numSamples_Integer, lStart__, x__, n_Integer, aStepSize_] :=
  NestList[fTakeStepMH[#, x, n, aStepSize] &, lStart, numSamples]

n = 8000;
lSamples1 =
  Flatten[ParallelTable[
    fMetropolisHastings[n, RandomReal[{1, 2}, 2], newdata, 100, 0.5][[
      n/2 ;;]], {i, 1, 12, 1}], 1];

SmoothDensityHistogram[lSamples1, 0.5, PlotRange -> {{0, 2}, {0, 20}},
  Mesh -> 5, Evaluate@options3, FrameLabel -> {"\[Alpha]", "\[Beta]"}]
```

For the reparameterised model, used to allow us to do vanilla Metropolis, it is essential to include the Jacobian of the transformation in the new prior. In this case the Jacobian ends up being $e^{\alpha_1}e^{\beta_1}$ where $\alpha_1 = log(\alpha)$ and $\beta_1 = log(\beta)$. The Mathematica code for this is,

```
fLikelihood2[\[Alpha]1_, \[Beta]1_, x__, n_Integer] :=
  Likelihood[
    BetaBinomialDistribution[Exp[\[Alpha]1], Exp[\[Beta]1], n], x]

fPrior2[\[Alpha]1_, \[Beta]1_] :=
  PDF[NormalDistribution[0, 1], \[Alpha]1] PDF[
  NormalDistribution[0, 1], \[Beta]1] (Exp[\[Alpha]1] Exp[\[Beta]1])

fPosterior2Unnormalised[\[Alpha]1_, \[Beta]1_, x__, n_Integer] :=
  fLikelihood2[\[Alpha]1, \[Beta]1, x, n] fPrior2[\[Alpha]1, \[Beta]1]

fStep1[\[Alpha]1_, \[Beta]1_, aStepSize_] :=
  RandomVariate[
    MultinormalDistribution[{\[Alpha]1, \[Beta]1},
    aStepSize IdentityMatrix[2]], {1}][[1]]

fAccept1[lCurrent__, lProposed__, x__, n_Integer] :=
  Module[{r =
    fPosterior2Unnormalised[lProposed[[1]], lProposed[[2]], x, n]/
    fPosterior2Unnormalised[lCurrent[[1]], lCurrent[[2]], x, n],
    aRand = RandomReal[]}, If[r > aRand, lProposed, lCurrent]]

fTakeStep1[lCurrent__, x__, n_Integer, aStepSize_] :=
  Module[{lProposed = fStep1[lCurrent[[1]], lCurrent[[2]], aStepSize]},
    fAccept1[lCurrent, lProposed, x, n]]

fMetropolis1[numSamples_Integer, lStart__, x__, n_Integer, aStepSize_] :=
  NestList[fTakeStep1[#, x, n, aStepSize] &, lStart, numSamples]

fMetropolisTransformed[numSamples_Integer, lStart__, x__,
                       n_Integer, aStepSize_] :=
  Module[{lSample =
    fMetropolis1[numSamples, lStart, x, n, aStepSize]}, {Exp[#[[1]]],
      Exp[#[[2]]]} & /@ lSample]

n = 8000;
lSamples = ParallelTable[fMetropolisTransformed[n,
  RandomVariate[
    MultinormalDistribution[{0, 0}, IdentityMatrix[2]], {1}][[1]],
      newdata, 100, 0.05][[n/2 ;;]], {i, 1, 12, 1}];

SmoothDensityHistogram[Flatten[lSamples, 1], 0.5,
  PlotRange -> {{0, 2}, {0, 20}}, Mesh -> 5, Evaluate@options3,
  FrameLabel -> {"\[Alpha]", "\[Beta]"}]
```

Problem 13.1.20. Construct 80% credible intervals for the parameters of the beta-binomial distribution.

The posteriors for this example are shown in Figure 13.10. I actually found it easier to generate the quantiles via sampling here, and found that the 80% credible intervals were approximately:

$$0.59 \le \alpha \le 1.75$$
$$5.68 \le \beta \le 12.95$$

Problem 13.1.21. Carry out appropriate posterior predictive checks using the new model. How does it fare?

I used my sampled parameters here to simulate posterior predictive data (see Figure 13.11). The new posterior predictive data now encompasses the range seen in the actual data. Hence we can be more content with using this new model.

13.2 The fairground revisited

You again find yourself in a fairground, and where there is a stall offering the chance to win money if you participate in a game. Before participating you watch a few other plays of the game (by other people in the crowd) to try to determine whether you want to play.

Problem 13.2.1. In the most-boring version of the game, a woman flips a coin and you bet on its outcome. If the coin lands heads-up, you win; if tails, you lose. Based on your knowledge of similar games (and knowledge that the game must be rigged for the woman to make a profit!) you assume that the coin must be biased towards tails. As such you decide to specify a prior on the probability of the coin falling heads-up as $\theta \sim beta(2, 5)$. Graph this function, and – using your knowledge of the beta distribution – determine the mean parameter value specified by this prior.

The prior can be graphed using,

```
curve(dbeta(theta, 2, 5), xname='theta', xlab='theta', ylab='pdf')
```

whose mean is 2/7.

Problem 13.2.2. You watch the last 10 plays of the game, and the outcome is heads 3/10 times. Assuming a binomial likelihood, create a function that determines the likelihood for a given value of the probability of heads, θ. Hence or otherwise, determine the maximum likelihood estimate of θ.

The function is given below,

```
fLikelihood <- function(Z, theta, N){
  return(dbinom(Z, N, theta))
}
curve(fLikelihood(3, theta, 10), 0, 1, xname="theta", xlab='theta',
      ylab='likelihood')
```

The likelihood is peaked at 0.3 – the maximum likelihood estimate of the parameter value.

Problem 13.2.3. Graph the likelihood $\times$ prior. From the graph approximately determine the MAP θ estimate value.

This can be done using the previously-created likelihood function and the below function,

```r
fLikelihoodTimesPrior <- function(Z, theta, N, a, b){
  return(fLikelihood(Z, theta, N) * dbeta(theta, a, b))
}

curve(fLikelihoodTimesPrior(3, theta, 10, 2, 5),
      xname='theta', xlab='theta')
```

which is peaked around 0.27. This can be determined with the below, although is not necessary for the approximate nature of this question,

```r
optim(0.5, function(theta) -fLikelihoodTimesPrior(3, theta, 10, 2, 5),
      lower=0, upper=1, method="L-BFGS-B")
```

Note the "−" is necessary in the above because `optim` does minimisation by default.

Problem 13.2.4. By using R's `integrate` function find the denominator, and hence graph the posterior pdf.

The denominator can be found by the below,

```r
integrate(function(theta) fLikelihoodTimesPrior(3, theta, 10, 2, 5), 0, 1)
```

which is approximately 0.164. Hence the posterior graph is just,

```r
fPosterior <- function(Z, theta, N, a, b){
  aInt = integrate(function(theta1)
                 fLikelihoodTimesPrior(Z, theta1, N, a, b), 0, 1)[[1]]
  return((1 / aInt) * fLikelihood(Z, theta, N) * dbeta(theta, a, b))
}

curve(fPosterior(3, theta, 10, 2, 5), 0, 1, xname = 'theta',
      xlab = 'theta', ylab = 'pdf')
```

Problem 13.2.5. Use your posterior to determine your break-even/fair price for participating in the game, assuming that you win £1 if the coin comes up heads, and zero otherwise.

This is just the mean of the posterior. Using conjugate prior rules, the posterior is a beta(2+3,5+10-3) distribution, which has a mean of £5/17$\approx$ 0.29. Alternatively, using your posterior, and R's numeric integration,

```r
integrate(function(theta) theta * fPosterior(3, theta, 10, 2, 5), 0, 1)
```

which should give the same answer.

Problem 13.2.6. Another variant of the game is as follows: the woman flips a first coin – if it is tails you lose ($Y_i = 0$), and if it is heads you proceed to the next step. In this step, the woman flips another coin ten times, and records the number of heads, Y_i, which equals your winnings. Explain why a reasonable choice for the likelihood might be,

$$L(\theta,\phi|Y_i) = \begin{cases} (1-\theta)+\theta(1-\phi)^{10}, & \text{if } Y_i = 0 \\ \theta\binom{10}{Y_i}\phi^{Y_i}(1-\phi)^{10-Y_i}, & \text{if } Y_i > 0 \end{cases}$$

where θ and ϕ are the probabilities of the first and second coins falling heads-up, and Y_i is the score on the game.

Problem 13.2.7. Using the above formula, write down the overall log-likelihood for a series of N observations for $Y_i = (Y_1, Y_2, ..., Y_N)$.

Assuming conditional independence of the observations, the overall likelihood is given by,

$$L(\theta,\phi|Y_1,Y_2,...Y_N) = \prod_{Y_i=0}\left[(1-\theta)+\theta(1-\phi)^{10}\right]\prod_{Y_i>0}\theta\binom{10}{Y_i}\phi^{Y_i}(1-\phi)^{10-Y_i} \tag{13.4}$$

$$= \left[(1-\theta)+\theta(1-\phi)^{10}\right]^{N_{Y_i=0}}\times\prod_{Y_i>0}\theta\binom{10}{Y_i}\phi^{Y_i}(1-\phi)^{10-Y_i} \tag{13.5}$$

where $N_{Y_i=0}$ is the number of times that $Y_i = 0$. Hence the log-likelihood is given by,

$$\log L(\theta,\phi|Y_1,Y_2,...Y_N) = N_{Y_i=0}\log\left[(1-\theta)+\theta(1-\phi)^{10}\right] + N_{Y_i>0}\log\theta + \sum_{Y_i>0}\log\binom{10}{Y_i}\phi^{Y_i}(1-\phi)^{10-Y_i} \tag{13.6}$$

Problem 13.2.8. Using R's `optim` function determine the maximum likelihood estimate of the parameters for $Y_i = (3, 0, 4, 2, 1, 2, 0, 0, 5, 1)$.

Hint 1: Since R's `optim` function does minimisation by default, you will need to put a minus sign in front of the function to maximise it.

First of all we need a function that determines the log likelihood,

```
fLogLikelihoodHarderAll <- function(lY, theta, phi){
  N0 <- sum(lY == 0)
  N1 <- sum(lY > 0)
  lY1 <- lY[lY > 0]
  aLogLikelihood <- N0 * log((1 - theta) + theta * (1 - phi) ^ 10) +
                    N1 * log(theta) +
                    sum(sapply(lY1, function(Y) log(choose(10, Y) * phi ^ Y *
                                                    (1 - phi) ^ (10 - Y))))
  return(aLogLikelihood)
}
```

which we then use as an argument to,

```
lY <- c(3,0,4,2,1,2,0,0,5,1)
optim(c(0.2, 0.2),
      function(theta) -fLogLikelihoodHarderAll(lY, theta[1], theta[2]),
      lower = c(0.001, 0.001), upper=c(0.999, 0.999), method="L-BFGS-B")
```

where we have avoided the infinities by using bounds that are a bit away from the edge of the domain. The parameter estimates from this are $\theta \approx 0.75$ and $\phi \approx 0.24$.

Problem 13.2.9. Determine confidence intervals on your parameter estimates. (Hint 1: use the second derivative of the log-likelihood to estimate the Fischer Information matrix, and hence determine the Cramer-Rao lower bound. Hint 2: use Mathematica.)

Used Mathematica to do the differentiation here (I know this is cheating...).

```
fLogLikelihood[lY__, \[Theta]_, \[Phi]_] := Block[{N0 = Count[lY, 0],
  N1 = Count[lY, _?Positive],
  lY1 = Cases[lY, _?Positive]},
  N0 Log[(1 - \[Theta]) + \[Theta] (1 - \[Phi])^10] +
  N1 Log[\[Theta]] +
  Sum[Log[Binomial[10, Y] \[Phi]^Y (1 - \[Phi])^(10 - Y)], {Y, lY1}]]

fInformationMatrix[lY__, \[Theta]_, \[Phi]_] :=
  -D[fLogLikelihood[lY, \[Theta]1, \[Phi]1], {{\[Theta]1, \[Phi]1},
  2}] /. {\[Theta]1 -> \[Theta], \[Phi]1 -> \[Phi]}

fCRLB[lY_] := Block[{lParams =
  NMaximize[{fLogLikelihood[lY, \[Theta]1, \[Phi]1],
  0 <= \[Theta]1 <= 1, 0 <= \[Phi]1 <= 1},
  {\[Theta]1, \[Phi]1}][[2]], \[Theta], \[Phi]}, {\[Theta], \
  \[Phi]} = {\[Theta]1, \[Phi]1} /. lParams;
  1/Length@lY Inverse@fInformationMatrix[lY, \[Theta], \[Phi]]]
```

which when evaluated on the data yields a lower bound on the variances: $var(\theta) \approx 0.0025$ and $var(\phi) \approx 0.0003$, which are then used to calculate standard deviations: 0.05 and 0.02 for θ and ϕ respectively. Therefore the approximate 95% confidence intervals (based on the asymptotic normal approximation – multiply std. deviations by 1.96) are, $0.65 \leq \theta \leq 0.85$ and $0.21 \leq \phi \leq 0.27$.

Problem 13.2.10. Assuming uniform priors for both θ and ϕ create a function in R that calculates the unnormalised posterior (the numerator of Bayes' rule).

This is straightforward, since the priors are both unity, the numerator of Bayes' rule is just the likelihood. This function needs to be written in R however,

```
fLikelihoodHarderAll <- function(lY, theta, phi){
  N0 <- sum(lY == 0)
  N1 <- sum(lY > 0)
  lY1 <- lY[lY > 0]
  aLikelihood <- ((1 - theta) + theta * (1 - phi) ^ 10) ^ N0 *
    prod(sapply(lY1,
         function(Y) theta * choose(10, Y) * phi^Y * (1 - phi) ^ (10 - Y)))
  return(aLikelihood)
}

fUnnormalisedPosterior <- function(lY, theta, phi){
  return(fLikelihoodHarderAll(lY, theta, phi))
}
```

Problem 13.2.11. By implementing the Metropolis algorithm, estimate the posterior means of each parameter. (Hint 1: use a normal proposal distribution. Hint 2: use periodic boundary conditions on each parameter, so that a proposal off one side of the domain maps onto the other side.)

The various functions that are necessary to implement the sampling here are,

```
fProposal <- function(theta, phi, sigma){
  theta.prop <- rnorm(1, theta, sigma)
  phi.prop <- rnorm(1, phi, sigma)
  theta.prop <- theta.prop %% 1
  phi.prop <- phi.prop %% 1
  return(list(theta.prop=theta.prop, phi.prop=phi.prop))
}

fProposeAndAccept <- function(lY, theta, phi, sigma){
  lProposed <- fProposal(theta,phi,sigma)
  theta.prop <- lProposed$theta.prop
  phi.prop <- lProposed$phi.prop
  aCurrent <- fUnnormalisedPosterior(lY, theta, phi)
  aProposed <- fUnnormalisedPosterior(lY, theta.prop, phi.prop)
  r = aProposed / aCurrent
  if (r > runif(1)){
    theta.new = theta.prop
    phi.new = phi.prop
  }
  else{
    theta.new = theta
    phi.new = phi
  }
  return(list(theta = theta.new, phi=phi.new))
}
```

```
fMetropolis <- function(numIterations, lY, theta.start, phi.start, sigma){
  lTheta <- vector(length=numIterations)
  lPhi <- vector(length=numIterations)
  lTheta[1] <- theta.start
  lPhi[1] <- phi.start
  for(i in 2:numIterations){
    lParams <- fProposeAndAccept(lY, lTheta[i - 1], lPhi[i - 1], sigma)
    lTheta[i] <- lParams$theta
    lPhi[i] <- lParams$phi
  }
  return(list(theta=lTheta, phi=lPhi))
}
```

The sampler can be run for 100,000 samples, and a 2d density plot used to visualise the posterior samples using,

```
lSamples <- fMetropolis(100000, lY, 0.5, 0.5, 0.1)

library(ggplot2)
aDF <- data.frame(theta=lSamples$theta, phi=lSamples$phi)
ggplot(aDF, aes(x=theta, y=phi)) + geom_point() + geom_density2d()
```

and you should see much of the posterior weight around the maximum likelihood estimates we obtained previously. The posterior means are about 0.71 and 0.25 respectively.

Problem 13.2.12. Find the 95% credible intervals for each parameter.

This is easily done using the quantile function,

```
quantile(lSamples$theta, 0.025)
quantile(lSamples$theta, 0.975)
quantile(lSamples$phi, 0.025)
quantile(lSamples$phi, 0.975)
```

and you should get something like $0.42 \leq \theta \leq 0.96$ and $0.15 \leq \phi \leq 0.36$.

Problem 13.2.13. Using your posterior samples determine the fair price of the game. (Hint: find the mean of the posterior predictive distribution.)

A function that implements the game is,

```
fGenerateData <- function(N, theta, phi){
  lY <- vector(length=N)
  for(i in 1:N)
    lY[i] <- ifelse(theta > runif(1), rbinom(1, 10, phi), 0)
  return(lY)
}
```

Now all we do is feed in the respective values of θ and ϕ,

```
lY.posteriorPredictive <- vector(length=length(lSamples$theta))
for(i in 1:length(lSamples$theta)){
  lY.posteriorPredictive[i] <- fGenerateData(1, lSamples$theta[i],
                                             lSamples$phi[i])
}
hist(lY.posteriorPredictive)
mean(lY.posteriorPredictive)
```

which is about £1.75.

13.3 Malarial mosquitoes

Suppose that you work for the WHO where it is your job to research the behaviour of malaria-carrying mosquitoes. In particular, an important part of your research remit is to estimate adult mosquito lifespan. The lifespan of an adult mosquito is a critical determinant of the severity of malaria, since the longer a mosquito lives the greater the chance it has of a. becoming infected by biting an infected human; b. surviving the period where the malarial parasite undergoes a metamorphosis in the mosquito gut and migrates to the salivary glands; and c. passing on the disease by biting an uninfected host.

Suppose you estimate the lifespan of mosquitoes by analysing the results of a mark-release-recapture field experiment. The experiment begins with the release of 1000 young adult mosquitoes (assumed to have an adult age of zero); each of which has been marked with a fluorescent die. On each day (t) you attempt to collect mosquitoes using a large number of traps, and count the number of marked mosquitoes that you capture (X_t). The mosquitoes caught each day are then re-released unharmed. The experiment goes on for 15 days in total.

Since X_t is a count variable and you assume that the recapture of an individual marked mosquito is i.i.d., then you choose to use a Poisson model (as an approximation to the binomial since n is large):

$$X_t \sim Poisson(\lambda_t)$$
$$\lambda_t = 1000 \times exp(-\mu t)\psi$$

where μ is the mortality hazard rate (assumed to be constant) and ψ is the daily recapture probability. You use a $\Gamma(2,20)$ prior for μ (which has a mean of 0.1), and a beta(2,40) prior for ψ.

The data for the experiment is contained in the file `RWM_mosquito.csv`.

Problem 13.3.1. Using the data create a function that returns the likelihood. (Hint: it is easiest to first write a function that accepts (μ, ψ) as an input, and outputs the mean on a day t.)

The function that returns the mean on a given day is,

```r
fMean <- function(mu, psi, t){
  return(1000 * exp(-mu * t) * psi)
}

curve(fMean(0.1, 0.05, t), 0, 20, xname='t')
```

Now creating a function that returns the likelihood,

```r
fLikelihood <- function(mu, psi, lData){
  t <- lData$time
  X <- lData$recaptured
  lMean <- sapply(t, function(x) fMean(mu, psi, x))
  lLikelihood <- sapply(seq_along(t), function(i) dpois(X[[i]], lMean[[i]]))
  return(prod(lLikelihood))
}
```

Problem 13.3.2. Find the maximum likelihood estimates of (μ, ψ). (Hint 1: this may be easier if you create a function that returns the log-likelihood, and maximise this instead. Hint 2: use R's optim function.)

In Mathematica I found that ML estimates of $(\mu, \psi) = (0.097, 0.041)$. This was using the inbuilt NMaximise function which uses "Nelder-Meda" to find the maxima. To do this in R use,

```r
fLogLikelihood <- function(params, lData){
  mu <- params[1]
  psi <- params[2]
  t <- lData$time
  X <- lData$recaptured
  lMean <- sapply(t, function(x) fMean(mu, psi, x))
  lLikelihood <- log(sapply(seq_along(t), function(i) dpois(X[[i]], lMean[[i]])))
  return(sum(lLikelihood))
}

optim(c(0.2, 0.1), function(params) -fLogLikelihood(params, lData),
      lower = c(0.001,0.001), upper=c(1,1), method='L-BFGS-B')
```

Problem 13.3.3. Construct 95% confidence intervals for the parameters. (Hint: find the information matrix, and use it to find the Cramer-Rao lower bound. Then find approximate confidence intervals by using the Central Limit Theorem.)

The point of this question is its difficulty to some extent. I want people to see how difficult it is to derive approximate estimates of the uncertainty of a parameter in Frequentist analyses. To do this you first of all find an estimate of the information matrix - essentially the negative of the Hessian matrix of second derivatives - at the ML estimates we found in the previous part. You then find its inverse, and the square root of its diagonal elements are the estimates of the parameter's standard error. To convert this to a confidence interval I am using a normal approximation, meaning that we simply multiply the standard errors by 1.96 and add on to the parameter estimates. This results in the following:

$$0.07 \leq \mu \leq 0.12$$
$$0.033 \leq \psi \leq 0.049$$

Note these are approximate - I have used a normal approximation to derive these. This may not be particularly valid here since the parameters are close to zero. Also, note that these confidence intervals can contain negative values; to do things properly we should really transform to a unconstrained space then back to (0,1).

Problem 13.3.4. Write a function for the prior, and use this to create an expression for the un-normalised posterior.

Problem 13.3.5. Create a function that proposes a new point in parameter space using a log-normal proposal with mean at the current μ value, and a $beta(2+\psi, 40-\psi)$ proposal for ψ. (Hint: use a $log-\mathcal{N}(0.5(-\sigma^2+2log(\mu)),\sigma)$, where μ is the current value of the parameter.)

Problem 13.3.6. Create a function that returns the ratio of the un-normalised posterior at the proposed step location, and compares it to the current position.

Problem 13.3.7. Create a Metropolis-Hastings accept-reject function.

This isn't as trivial as for "vanilla" Metropolis - now we need to use the full Metropolis-Hastings accept-reject rule, which calculates the statistic:

$$r = \frac{p(\theta')}{p(\theta)} \times \frac{g(\theta|\theta')}{g(\theta'|\theta)} \tag{13.7}$$

where $g(\theta'|\theta)$ is the value of the PDF of the jumping kernel centred at current parameters, at the proposed parameter values. Since the two-proposal distributions are independent, we can find this just by multiplying together the log-normal and beta PDFs.

Problem 13.3.8. Create a Metropolis-Hastings sampler by combining your proposal and accept-reject functions.

Problem 13.3.9. Use your sampler to estimate the posterior mean of μ and ψ for a sample size of 4000 (discard the first 50 observations.) (Hint: if possible, do this by running 4 chains in parallel.)

The reject rate is pretty high here (probably because our proposal distribution takes no account of the posterior geometry), meaning that we are inefficient at exploring the posterior. However, after about 4000 samples we get a reconstructed posterior that looks similar to the exact one (Figure 13.12). The 80% credible intervals I obtain on each parameter are:

$$0.08 \leq \mu \leq 0.12$$
$$0.034 \leq \psi \leq 0.048$$

These are pretty similar to the approximate (95%) confidence intervals I obtained above.

Problem 13.3.10. By numeric integration compute numerical estimates of the posterior means of μ and ψ. How does your sampler's estimates compare with the actual values? How do these compare to the MLEs?

The exact values are $(\mu, \psi) = (0.096, 0.40)$ both of which lie right in the middle of the sampled credible intervals $\implies$ the sampler does a pretty good job here! The MLEs also look similar because we are using fairly wide priors here.

Problem 13.3.11. Carry out appropriate posterior predictive checks to test the fit of the model. What do these suggest might be a more appropriate sampling distribution? (Hint: generate a single sample of recaptures for each value of (μ, ψ) using the Poisson sampling distribution. You only need to do this for about 200 sets of parameter values to get a good idea.)

From the actual versus simulated data series it is evident that the posterior predictive samples do not reproduce the degree of variation we see in the data (Figure 13.13). In particular there are a number of days (2,7,8,10) where the actual recaptured value lies outside the posterior predictive range. This is because the assumption of independent recaptures, upon which the Poisson model is based, is likely violated. Intuitively individual mosquitoes will respond similarly to fluctuations in weather, which may cause them to be recaptured in "clumps". This lack of independence in the recaptures causes over-dispersion in the recapture data.

A more appropriate model that allows for the non-independence in recaptures is the negative binomial likelihood. This model becomes a Poisson distribution in the limit $\kappa \rightarrow \infty$, where $\frac{1}{\kappa}$ represents the degree of over-dispersion seen in the data.

Problem 13.3.12. An alternative model that incorporates age-dependent mortality is proposed where:

$$\lambda_t = 1000 \times exp(-\mu t^{\beta+1})\psi \tag{13.8}$$

where $\beta \geq 0$. Assume that the prior for this parameter is given by $\beta \sim exp(5)$. Using the same log-normal proposal distribution as for μ create a Random Walk Metropolis sampler for this new model. Use this sampler to find 80% credible intervals for the (μ, ψ, β) parameters.

The 80% credible intervals I obtained for the parameters were:

$$0.049 \leq \mu \leq 0.089$$
$$0.034 \leq \psi \leq 0.043$$
$$0.065 \leq \beta \leq 0.194$$

Problem 13.3.13. Look at a scatter plot of μ against β. What does this tell you about parameter identification in this model?

There is strong negative correlation between these parameter estimates (Figure 13.14). This suggests that it may be difficult to disentangle the effects of one parameter from another one. This makes intuitive sense; if $\mu \uparrow$ then $\beta \downarrow$ to allow lifespan to stay roughly constant.

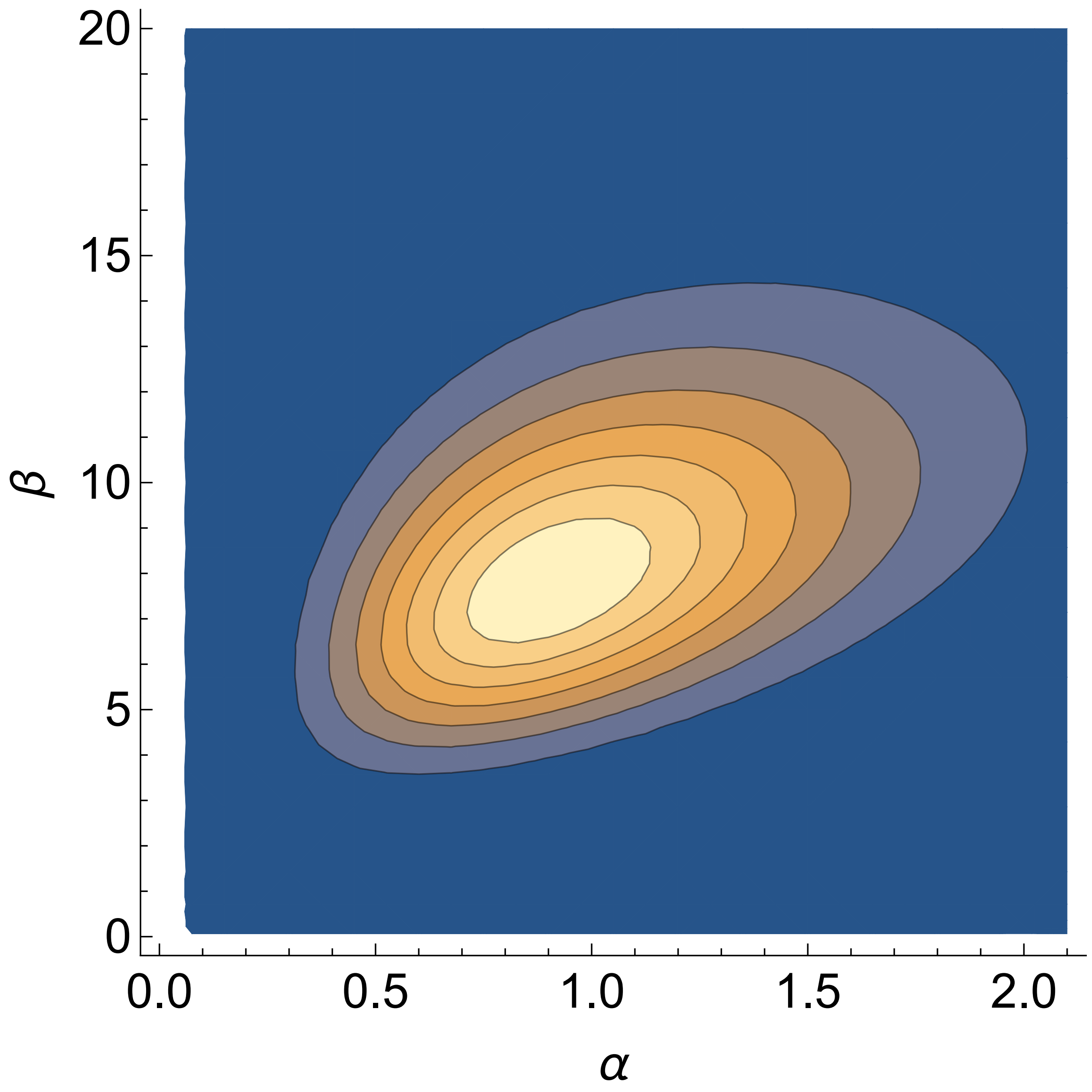

Figure 13.9: The joint posterior for the beta-binomial parameters.

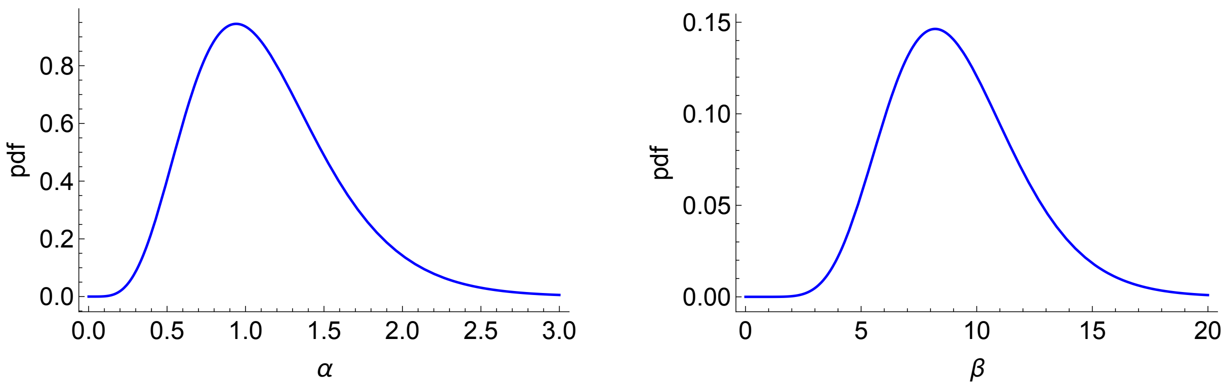

Figure 13.10: The posteriors for the beta-binomial parameters.

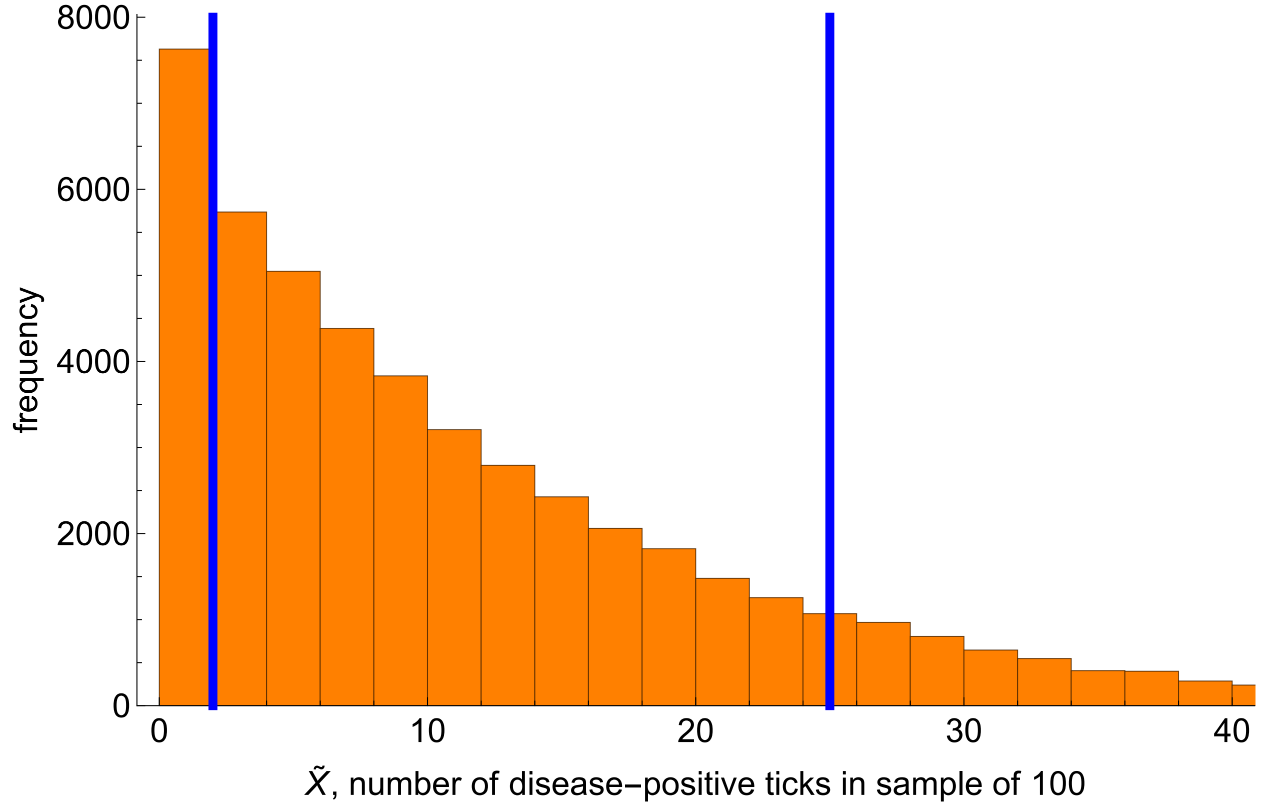

Figure 13.11: The posterior predictive distribution for the beta-binomial sampling model.

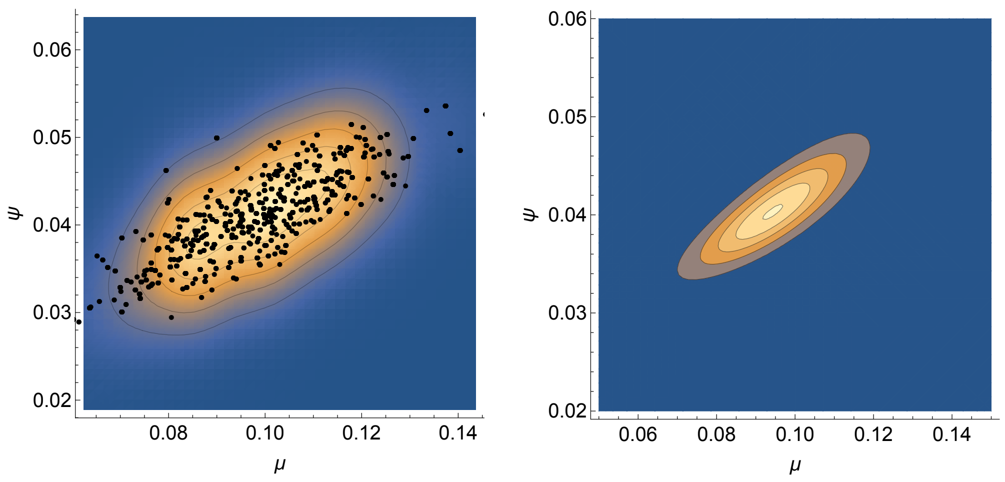

Figure 13.12: The sample-reconstructed posterior (left) versus the true posterior (right) for the mosquito question, where we assume a constant mortality rate.

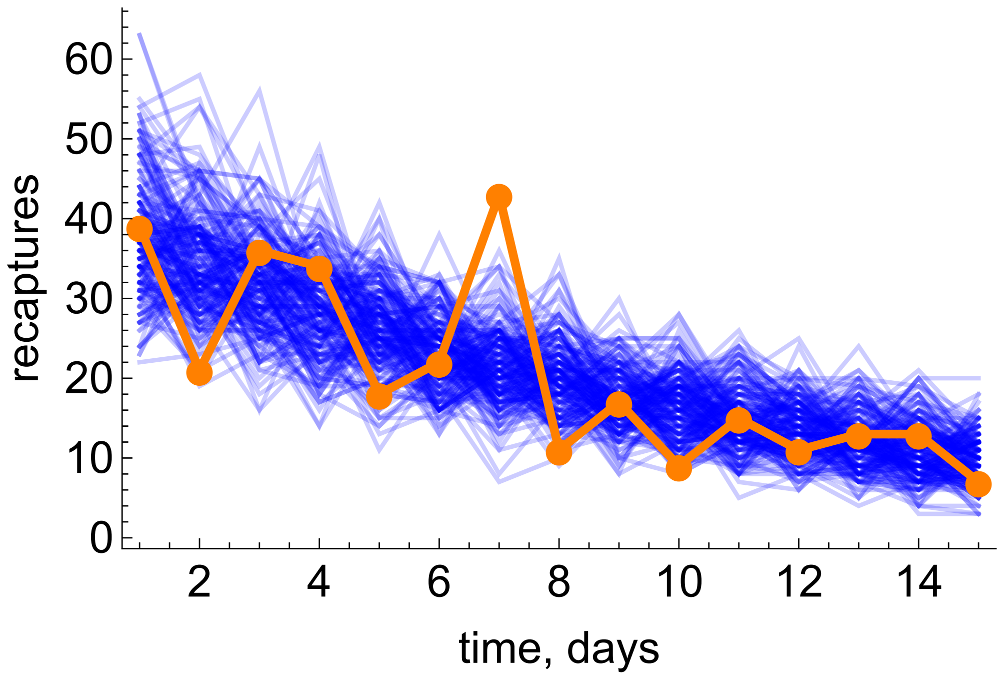

Figure 13.13: Samples from the posterior predictive distribution (blue) versus the actual recaptures (orange).

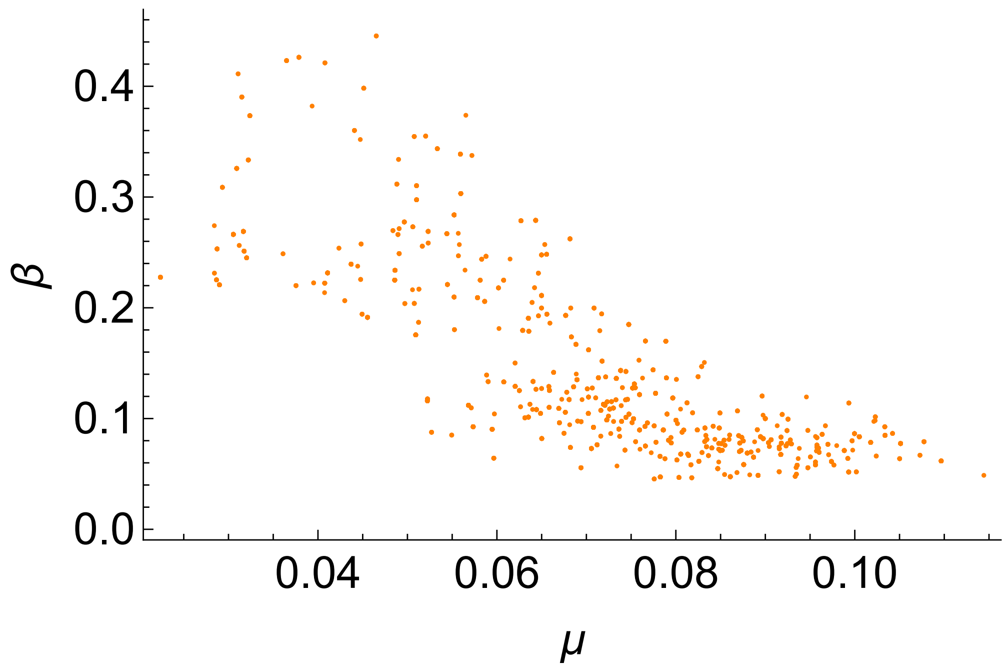

Figure 13.14: Posterior samples from (μ, β) for the mosquito model that incorporates age-dependent mortality.

Bibliography