4

BAYESIAN STATISTICS

"Probability does not exist."
—Bruno de Finetti

Classical statistics involves ways to test hypotheses and estimate confidence intervals. Bayesian statistics involves methods to calculate probabilities associated with your hypotheses. The result is a posterior distribution that combines information from your data with prior beliefs. The term "Bayesian" comes from Bayes's theorem—a single formula for obtaining the posterior distribution. It is the cornerstone of modern statistical methods.

In this chapter we introduce Bayesian statistics. We begin by considering the problem of learning about the population proportion (percentage) of all hurricanes that make landfall. We then consider the problem of estimating how many days it takes your manuscript to get accepted for publication. Again, we encourage you to follow along by typing the code into your R console.

4.1 LEARNING ABOUT THE PROPORTION OF LANDFALLS

Models that have skill at forecasting hurricane counts can be made relevant to society if they include an estimate of the proportion of those that make landfall. Before examining the data, you hold a belief about the value of this proportion. You model your belief in terms of a prior distribution. Then after examining some data, you update your belief about the proportion by computing the posterior distribution (Albert, 2009).

This setting allows you to predict the likely outcomes of a new sample taken from the population, for example, the proportion of landfalls for next year. The use of the pronoun "you" focuses attention on the Bayesian viewpoint that probabilities, [illegible] not exist but are how much *you* personally believe that something is true. Said another way, probabilities are subjective and based on all the relevant information available to you.

Here you think of a population consisting of past and future hurricanes in the North Atlantic. Then let π represent the proportion of this population that hit the United States at hurricane intensity. The value of π is unknown. You are interested in learning about what the value of π could be.

Bayesian statistics requires that you represent your belief about the uncertainty in this percentage with a probability distribution. The distribution reflects your subjective prior opinion about plausible values of π. Before you examine a sample of hurricanes, you think about what the value of π might be. If all hurricanes make landfall π is one, and if none make landfall π is zero. This is the nature of proportions, bounded below by zero and above by one. Also while the count of hurricanes is an integer the percentage that make landfall is a real number.

As a climatologist you are also aware that not all hurricanes make it to land. The nature of percentages together with your background provides you with information about π that is called "your prior." From this information, suppose you believe that the percentage of hurricanes making landfall in the United States is equally likely to be smaller or larger than 0.3. Moreover, suppose you are 90 percent confident that π is less than 0.5.

A convenient family of densities for a proportion is the beta. The beta density, here representing your prior belief about the population percentage of all hurricanes making landfall π, is proportional to

$$g(\pi) \propto \pi^{a-1}(1-\pi)^{b-1} \tag{4.1}$$

where the parameters a and b are chosen to reflect your prior beliefs about π.

The mean of a beta density is $m = a/(a+b)$ and the variance is $v = m(1-m)/(a+b+1)$. Unfortunately, it is difficult to assess values of m and v for distributions like the beta that are not symmetric. It is easier to obtain a and b indirectly through statements about the percentiles of the distribution. Recall from before you have a belief about the median (.3) and the 90th percentile (.5).

The `beta.select` function in the **LearnBayes** package (Albert, 2011) is useful for finding the two parameters (shape and scale) of the beta density that match this prior knowledge. The inputs are two lists, `q1` and `q2`, that define these two percentiles, and the function returns values for the beta parameters, a and b as respective elements of a vector.

```
> require(LearnBayes)
> q1 = list(p=.5, x=.3)
> q2 = list(p=.9, x=.5)
> beta.select(q1, q2)
[1] 3.26 7.19
```

Note the argument `p` is the distribution percentile and the argument `x` is a value for π.

Now you have your prior specified as a continuous distribution. You should plot your prior using the curve function to see if it looks reasonable relative to your beliefs.

```
> a = beta.select(q1, q2)[1]
> b = beta.select(q1, q2)[2]
> curve(dbeta(x, a, b), from=0, to=1,
+   xlab="Proportion of Landfalls",
+   ylab="Density", lwd=4, las=1, col="green")
> abline(v=.5, lty=2)
> abline(v=.3, lty=2)
```

As seen in Figure 4.1, the distribution appears to adequately reflect your prior knowledge of landfall percentages. For reference, the vertical dashed lines are the values you specified for the median and the 90th percentile. The abline function is used after the curve is plotted. The v argument in the function specifies where along the horizontal axis to draw the vertical line on the graph. Recall, to learn more about the function use ?abline.

This is a start, but you need to take your prior distribution and combine it with a likelihood function. The likelihood function comes from your data. It is a function that describes how likely it is, given your model, that you observed the data you actually have. It might sound a bit confusing, but consider that if you have a good model for your data, then the probability that your model will replicate your data should be relatively high.

For example, consider the number of hurricanes over the most recent set of years. Read the data files containing the annual basin-wide (A) and landfall (US) counts and compute the sum.[1]

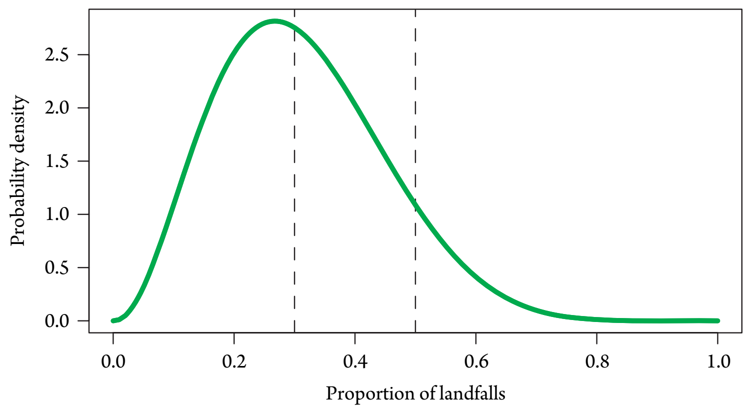

Figure 4.1 Beta density describing hurricane landfall proportion.

[1] The basin counts were obtained from http://www.aoml.noaa.gov/hrd/tcfaq/E11.html on January 17, 2012.

```
> A = read.table("ATL.txt", header=TRUE)
> US = read.table("H.txt", header=TRUE)
> Yr = 2006
> sum(A$H[A$Year >= Yr])
[1] 34
> sum(US$All[US$Year >= Yr])
[1] 4
```

You find 4 of the 34 hurricanes made landfall in the United States since 2006. You could simply report that $\pi = 0.12$ or 12 percent of hurricanes make landfall. However, this is the single sample estimate and it does not consider your prior belief about p. It also does not have an estimate of uncertainity. In addition, you might be interested in predicting the number of landfalls in a new sample of next year's hurricanes.

If you regard a "success" as a hurricane making landfall and you take a random sample of n hurricanes with s successes and $f = n - s$ failures, then the *likelihood function* (or simply, the likelihood) is given by

$$L(s,f|\pi) \propto \pi^s(1-\pi)^f, \qquad 0 < \pi < 1 \tag{4.2}$$

The likelihood depends on the model parameters, here only π and is defined by your observations written L(data|π). Specifically, the likelihood of π given that you observed four landfalls in a set of 34 hurricanes is equal to the probability of that particular observed set of outcomes.

Here the model is the beta density with parameters $a = s + 1$ and $b = f + 1$. You can convince yourself that the likelihood is maximized when $\pi = 0.12$ for $s = 4$ and $f = 30$ by plotting $L(4, 30, \pi)$ for π from 0 to 1.

Then with a likelihood function describing your data along with your prior beliefs, your posterior density for π is obtained up to a proportionality constant, by multiplying the prior density $(g(\pi))$ by the likelihood $(L(\text{data}|\pi))$, which is Bayes's rule given as

$$g(\pi|\text{data}) \propto g(\pi)L(\text{data}|\pi) \tag{4.3}$$

Your prior represents your thinking about the parameter (in terms of probability) of interest before examining data. Afterward, you have a likelihood function representing the probability of your data, given values for the parameter. Finally, the posterior probability distribution is computed using Bayes's rule.

It can be shown that if the prior is a beta density with parameters a and b and the likelihood is a beta density with parameters s and f, then the posterior density is also a beta density with parameters $a + s$ and $b + f$. This is an example of a conjugate model, where the prior and posterior densities belong to the same family of densities. To compute and plot the prior, likelihood, and posterior together, type

```
> s = sum(US$All[US$Year >= Yr])
> f = sum(A$H[A$Year >= Yr]) -
+   sum(US$All[US$Year >= Yr])
```

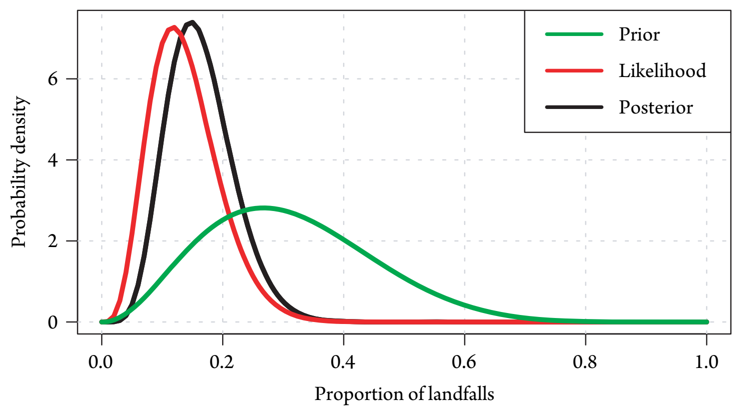

Figure 4.2 Densities describing hurricane landfall proportion.

```
> curve(dbeta(x, a + s, b + f), from=0, to=1,
+   xlab="Proportion of Landfalls", ylab="Density",
+   col=1, lwd=4, las=1)
> curve(dbeta(x, s + 1, f + 1), add=TRUE, col=2,
+   lwd=4)
> curve(dbeta(x, a, b), add=TRUE, col=3, lwd=4)
> legend("topright", c("Prior", "Likelihood",
+   "Posterior"), col=c(3, 2, 1), lwd=c(3, 3, 3))
```

The densities are shown in Figure 4.2. Note that the posterior density resembles the likelihood, but it is shifted in the direction of the prior. This is always the case. The posterior is a weighted average of the prior and the likelihood where the weights are proportional to the precision. The greater the precision the more weight it carries in determining the posterior.

For your prior, the precision is related to how committed you are to a particular value. Believing that there is a 90 percent chance that the proportion is less than 0.5 provides a quantitative level of commitment. The less committed you are, the flatter the prior distribution and the less weight it plays in the posterior. For your data, the precision is inversely related to the standard error, so directly related to the sample size. The more data you have, the more weight the likelihood plays in determining the posterior.

4.2 INFERENCE

A benefit of Bayesian statistics is that the posterior distribution contains the information you need to make all inferences. In this example, since the posterior is a beta distribution, the `pbeta` (cumulative distribution) and `qbeta` (quantile) functions can be used. For example, given your prior beliefs and your sample of data, how likely is it that the population percentage of landfalls is less than or equal to 25 percent? This is answered by computing the posterior probability $P(\pi \leq 0.25|\text{data, prior})$, which is given by

```
> pbeta(q=.25, a + s, b + f)
[1] 0.93
```

You interpret this probability in a natural way. You can state that given the evidence in hand (data and your prior belief), there is a 93 percent chance that less than or equal to a quarter of all hurricanes hit the United States (at hurricane intensity). Or you can state that it is quite unlikely (1.5%) that more than 3 in 10 hurricanes make it to the United States. (`1-pbeta(.3,a+s,b+f)`).

You cannot do this with classical statistics. Instead, the p-value resulting from a significance test is the evidence in support of a null hypothesis. Suppose the null hypothesis is that the population proportion exceeds 3 in 10. You state $H_o : \pi > 0.3$ against the alternative $H_a : \pi \leq 0.3$ and decide between the two on the basis of a p-value obtained by typing

```
> prop.test(s, s + f, p=.3, alt="less")$p.value
[1] 0.0165
```

where the argument `p` specifies the value of the null hypothesis and the argument `alt` specifies the direction of the alternative hypothesis.

The *o* null p-value of 0.016 indicates that if the null is true (the proportion is greater than 0.3), the evidence seems unusual. You conclude that there is moderate evidence against the null. However, it is incorrect to conclude from the p-value that there is only a 1.6 percent chance that the proportion of landfalls is greater than 0.3.

4.3 CREDIBLE INTERVAL

Bayesian interpretation extends to a posterior summaries. For instance, the 95 percent interval estimate for the percentage of landfalls is obtained from the 2.5th and 97.5th percentiles of the posterior density. This is done with the `qbeta` function by typing

```
> qbeta(c(.025, .975), a + s, b + f)
[1] 0.0713 0.2839
```

You state the 95 percent *credible interval* for the proportion is [0.071, 0.284] and conclude you are 95 percent confident that the true proportion lies inside this interval.

Note the distinction. With a credible interval you say that given the data and your prior beliefs, there is a 95 percent chance that population proportion lies within this particular interval. With a confidence interval you say you are 95 percent confident that the method produces an interval that contains the true proportion. In the former the population parameter is the random variable, and in the latter the interval is the random variable.

The 95 percent confidence interval estimate for the proportion π using a large enough sample of data, where $\hat{p}$ is the sample proportion and n is the sample size, is given by

$$\hat{p} \pm \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \times \Phi^{-1}(0.975) \tag{4.4}$$

where $\Phi^{-1}(0.975)$ (inverse of the cumulative distribution function) is the 97.5th percentile value from a standard normal distribution. The confidence interval is obtained by typing

```
> prop.test(s, s + f)$conf.int
[1] 0.0384 0.2839
attr(,"conf.level")
[1] 0.95
```

The result allows you to state that the 95 percent confidence interval for the proportion is [0.038, 0.284] and conclude that if you had access to additional samples and computed the interval in this way for each sample, 95 percent of the intervals will cover the true proportion.

In some cases, the intervals produced (confidence and credible) from the same data set are identical. They are almost certainly different if an informative prior is included and they may be different even if the prior is relatively uninformative. However, the interpretations are always different.

With Bayesian statistics inferential statements are easier to communicate. It is natural to talk about the probability that π falls within an interval or the probability that a hypothesis is true. Also Bayes's rule (Eq. 4.3) is the only thing you need to remember. It is used for small and large samples. It is the only consistent way to update your probabilities. The cost is that you need to specify your prior beliefs.

4.4 PREDICTIVE DENSITY

So far we focused on learning about the population proportion of hurricanes that make landfall. Suppose your interest is predicting the number of U.S. landfalls ($\tilde{l}$) in a future sample of seven hurricanes. If your current understanding of π is contained in the density $g(\pi)$ (e.g., the posterior density), then the predictive density for $\tilde{l}$ is given by

$$f(\tilde{l}) = \int f(\tilde{l}|\pi)g(\pi)d\pi \tag{4.5}$$

With a beta distribution, you can integrate to get an expression for the predictive density. The beta predictive probabilities are computed using the function `pbetap` from the **LearnBayes** package. The inputs are a vector `ab` containing the beta parameters, the size of the future sample m, and a vector of the number of future landfalls `lf`.

```
> ab = c(a + s, b + f)
> m = 7; lf = 0:m
> plf = pbetap(ab, m, lf)
> round(cbind(lf, plf), digits=3)
     lf   plf
[1,]  0 0.312
[2,]  1 0.367
[3,]  2 0.216
[4,]  3 0.081
[5,]  4 0.021
[6,]  5 0.004
[7,]  6 0.000
[8,]  7 0.000
```

Simulation provides a convenient way to compute a predictive density. For example, to obtain $\tilde{l}$, first simulate p from $g(p)$ and then simulate $\tilde{l}$ from the binomial distribution, $f_B(l|n,p)$. You can use this approach on your beta posterior. First simulate 1,000 draws from the posterior and store them in `p`.

```
> p = rbeta(n=1000, a + s, b + f)
```

Then simulate values of $\tilde{l}$ for these random values using the `rbinom` function and tabulate them.

```
> lc = rbinom(n=1000, size=7, prob=p)
> table(lc)
lc
  0   1   2   3   4   5
319 351 220  83  20   7
```

The table indicates that, of the 1,000 simulations, 319 of them resulted in no landfalls, 351 of them resulted in one landfall, and so on.

You save the frequencies in a vector and then convert them to probabilities by dividing by the total number of simulations. Finally plot the probabilities using the histogram argument (`type="h"`).

```
> freq = table(lc)
> pp = freq/sum(freq)
> plot(pp, type="h", xlab="Number of U.S. Hurricanes",
+    las=1, lwd=3, ylab="Predictive Probability")
```

The plot is shown in Figure 4.3. It is most likely that one of the seven hurricanes will hit the United States. The probability of four or more doing so is 2.7 percent.

The cumulative sum of the probabilities is found by typing

```
> cumsum(pp)
    0     1     2     3     4     5
0.319 0.670 0.890 0.973 0.993 1.000
```

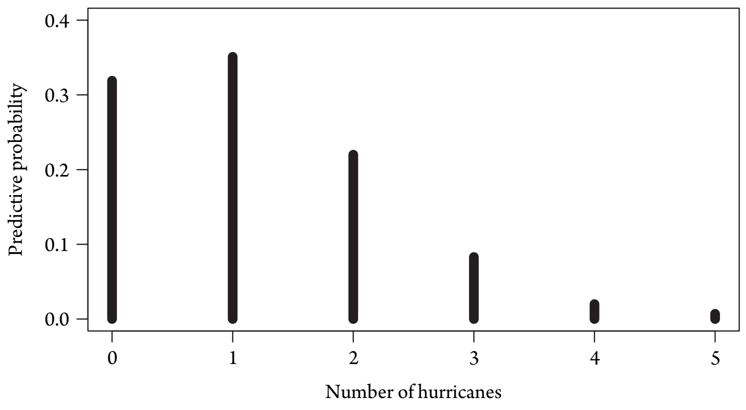

Figure 4.3 Predictive probabilities for the number of landfalling hurricanes.

The probability of three or fewer landfalls is 0.973. Suppose you wish to summarize this discrete predictive distribution by an interval that covers a certain amount of the probability.

This can be done using the function `discint` from the **LearnBayes** package. You combine the vector of probabilities with a vector of counts and specify a coverage probability as

```
> mc = length(pp) - 1
> int = discint(cbind(0:mc, pp), .95)
> int
$prob
    3
0.973

$set
0 1 2 3
0 1 2 3
```

The output has two lists, the minimum probability (`$prob`) greater than the specified coverage probability (here 95%) and the list of counts (`$set`) over which the individual probabilities are summed. You can see the probability is 97.3 percent that the number of U.S. hurricanes is one of these counts.

The interval covers a range of four counts, which is necessarily wider than the range of counts computed by using the bounds on the 97.3 percent credible interval around the population proportion. You check this by typing

```
> lp = (1 - as.numeric(int$prob))/2
> up = 1 - lp
> (qbeta(up, a + s, b + f) -
+   qbeta(lp, a + s, b + f)) * 7
[1] 1.67
```

This is because in predicting a specific value, as you saw in Chapter 3 with the prediction intervals from a linear regression model, there are two sources of uncertainty. Here the uncertainty about the population proportion π and the uncertainty about the count given an estimate of π.

4.5 IS BAYES'S RULE NEEDED?

Since any probabilities are ultimately subjective, why is Bayes's rule needed? Why not simply examine your data and subjectively determine your posterior probability distribution, updating what you already know? Would not this save you trouble? Strange as it sounds, it would actually cause you more trouble.

The catch is that it is hard to assess probabilities rationally. For example, probabilities must be nonnegative and sum to one. If these are not met, then your probabilities are said to be inconsistent. In fact they should obey certain axioms of coherent assessment (Winkler, 2003). For example, if you consider event A to be more likely than event B, and event B to be more likely than event C, then you should consider event A to be more likely than event C. If your probabilities fail to obey transitivity, then your assessment will be inconsistent in a decision-making sense.

This is not a serious impediment since you can easily remove your inconsistency once you are made aware of it. If someone notes an arithmetic error, you would certainly correct it. Similarly, if someone notes that your prior probabilities are not transitive, you would change them accordingly. In this regard, it is sometimes useful to attempt to assess the same set of probabilities in more than one way in order to "check" your assessments. You might assess probabilities using the mean and variance and then compare these to an assessment based on quantiles.

Importantly, if you obey the axioms of coherence, then to revise your probabilities you must use Bayes's rule. To do otherwise would be inconsistent. Numerous psychological studies have indicated that people do not always intuitively revise probabilities on the basis of this rule. People tend to give too little "weight" to the data. In general, this means that they do not change their probabilities as much as Bayes's theorem tells them that they should. By using the rule, you ensure consistency in updating your beliefs based on the evolving evidence.

4.6 BAYESIAN COMPUTATION

The models and examples presented in this chapter are reasonably simple: for example, inference for an unknown success probability, a proportion, the mean of a normal, and so on. For these problems, the likelihood functions are standard, and, if you use a conjugate model where the prior and posterior densities come from the same family, deriving the posterior density poses no computational burden. Indeed this is why conjugate models like the beta are widely used in Bayesian analysis (Jackman, 2009).

You showed earlier that the probability of a random hurricane hitting the United States can be modeled using a beta prior. In this case, computation is nothing but addition. But Bayesian computation quickly becomes more challenging when working with more complicated models or when you use nonconjugate models. Characterizing the posterior density in these cases becomes nontrivial. Many statistical models in geography and the social sciences have this feature (see the models in Chapter 12).

Options still exist. One is brute force. When the posterior distribution is complicated, you can compute values over a grid and then approximate the continuous posterior by a discrete one (Albert, 2009). This is possible for models with one or two parameters. In situations where you can directly sample from the posterior, a Monte Carlo (MC) algorithm gives an estimate for the posterior mean for any function of your parameters. In the situation where you cannot directly sample, you can use rejection sampling, provided you have a suitable proposal density or, in the most general case, you can adopt a Markov chain Monte Carlo (MCMC) approach.

4.6.1 Time-to-Acceptance

Here you assume that information about manuscript review time might be useful to authors and editors, especially if it can say something about future submissions. This motivates you to collect data on publication times from recent journals and to model them. Here you cannot exploit conjugacy so you approximate the posterior with a discrete distribution. You sample directly from the distribution and use an MC algorithm for summarizing your samples.

We used the American Meteorological Society (AMS) journals and the keyword "hurricane" appearing in published titles over the years 2008–2010 (see Hodges et al. (2012)). The search is done from the web site `http://journals.ametsoc.org/action/doSearch`. Selecting "Full Text" on a particular article brings up the abstract, keywords, and two dates: received and accepted—in month, day, year format.

The data were originally entered manually but are available to you by typing.

```
> art = read.csv("hurart.txt", header=TRUE)
```

For each article both dates are provided along with the lead author's last name and of the journal. There are 100 articles with the word "hurricane" in the title over the 3-year period. Of these 34, 41, and 25 had "accepted" years of 2008, 2009, and 2010, respectively. Articles appearing in 2008, but with accepted years before 2008 are not included.

Journals publishing these articles are shown in Figure 4.4. Ten different journals are represented. Thirty-seven of these articles are published in *Monthly Weather Review*, 22 in *Journal of the Atmospheric Sciences*, 13 in the *Journal of Climate*, and 12 in *Weather and Forecasting*.

First you compute the time period between received and accepted dates. This is done by converting the numeric values of year, month, and day to an actual date using the `ISOdate` function and then using the `difftime` to compute the time difference in days.

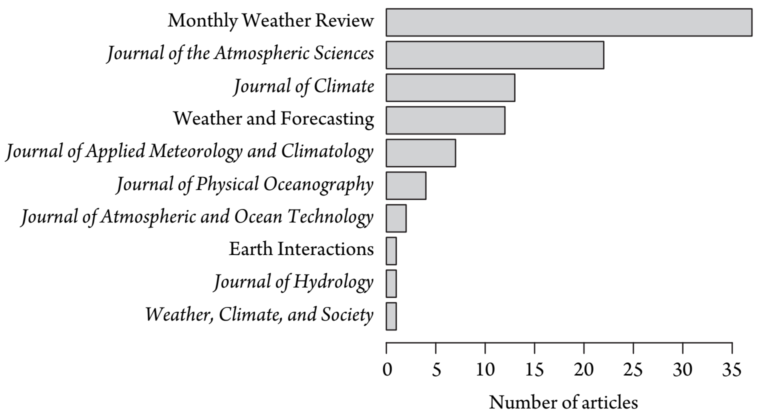

Figure 4.4 AMS journals publishing articles with 'hurricane' in the title.

```
> rec = ISOdate(art$RecYr, art$RecMo, art$RecDay)
> acc = ISOdate(art$FinYr, art$FinMo, art$FinDay)
> tau = difftime(acc, rec, units="days")
```

You call the temporal difference the time-to-acceptance (τ). This is your statistic of interest.

The mean τ is 193 in 2008 (type `mean(tau[art$FinYr==2008])`), 168 in 2009, and 193 in 2010. The mean τ for each of the four journals with the most articles is 166 (*Monthly Weather Review*), 169 (*Journal of the Atmospheric Sciences*), 189 (*Journal of Climate*), and 181 (*Weather and Forecasting*). On average, the *Journal of Climate* is slowest and *Monthly Weather Review* is fastest, but the difference is less than 3.5 weeks.

Undoubtedly, there are many factors that influence the value of τ. On the editor's side, there are prereview, review requests, and dispensation decisions among others. On the reviewer's side, there is workload as well as breaks for travel and vacation, and on the author's side, there is the effort needed to revise and resubmit.

The goal for you is a predictive distribution for τ. This will allow you to make inferences about the time-to-acceptance for your future manuscript submissions. You assume that is τ a random variable having a gamma density given by

$$f(\tau|\alpha,\beta) = \frac{\tau^{\alpha-1}\exp(\frac{-\tau}{\beta})}{\beta^{\alpha}\Gamma(\alpha)} \tag{4.6}$$

where α and β are the shape and scale parameters, respectively, and $\Gamma(\alpha) = (\alpha - 1)!$ The gamma density is commonly used to model time periods (wait times, phone call lengths, etc.). If you place a uniform prior distribution on the parameter vector (α, β), the posterior density is given by

$$g(\alpha,\beta|\tau) \propto g(\alpha,\beta)f(\tau|\alpha,\beta) \tag{4.7}$$

The uniform prior is consistent with a judgment that the parameters are the same regardless of author or journal. Random draws from this joint posterior density are summarized and used as parameters to a gamma density to draw predictive samples for τ.

To make the computation easier, the posterior density is reformulated in terms of $\log\alpha$ and $\log\mu$, where $\mu = \alpha \times \beta$ is the posterior mean of τ (Albert, 2009). The posterior density is available in the `gsp.R` file (from Jim Albert). Here you source the function by typing

```
> source("gsp.R")
```

Remember to include the quotes around the file name.

Given a pair of parameter values $(\log\alpha, \log\mu)$, the function computes the posterior probability given the data and the posterior density using

$$\log g(\alpha, \frac{\mu}{\alpha}|\tau) = \log\mu + \sum_{i=1}^{n} \log f(\tau_i|\alpha, \frac{\mu}{\alpha}) \tag{4.8}$$

It performs this computation using pairs of parameters defined as a two-dimensional grid spanning the domain of the posterior.

The `gsp` function is used to determine the posterior density as defined in Eq. 4.8 over a two-dimensional grid of parameter values (on log scales) using the `mycontour` function from the **LearnBayes** package. It is also used to sample from the posterior using the `simcontour` function from the same package.

Using these functions, you contour the joint posterior of the two parameters and then add 1,000 random draws from the distribution by typing

```
> lim = c(.8, 2.1, 5, 5.45)
> mycontour(gsp, limits=lim, data=tau,
+   xlab="log alpha", ylab="log mu")
> s = simcontour(gsp, lim, data=tau, m=1000)
> points(s$x, s$y, pch=19, col="gray", cex=.5)
```

These computations may take several seconds to complete. The result is shown in Figure 4.5. The contours from inside-out are the 10 percent, 1 percent, and 0.1 percent probability intervals on a log scale.

The graph shows that the mean time-to-acceptance is about 181 days (on the ordinate exp(5.2)). This might be useful to an editor. But the posterior gives additional information. For example, the editor can use your model to estimate the probability that the average time-to-acceptance will exceed 200 days for the set of manuscripts arriving next month. Since the model provides random values, the question is answered by finding the percentage of values exceeding the logarithm of 200. Assuming that the set of manuscripts is a random sample, the model predicts 4.3 percent.

Averages are not particularly useful to you as an author. You would like to know if a recently submitted manuscript will be accepted in less than 120 days. To answer this question, you take random draws of τ's from a gamma density using

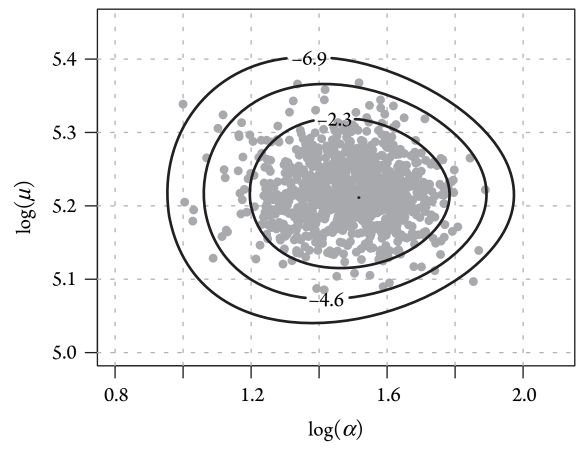

Figure 4.5 Posterior density of the gamma parameters for a model describing time-to-acceptance.

the random draws from the posterior parameter distribution. This is done with the `rgamma` function and then finding the percentage of these draws less than this many days.

```
> alpha = exp(s$x)
> mu = exp(s$y)
> beta = mu/alpha
> taum = rgamma(n=1000, shape=alpha, scale=beta)
```

Here the model predicts a probability of 26 percent. Note that this probability is lower than the average percentage less than 120 days as it includes additional uncertainty associated with modeling an individual estimate.

Changes to review rules and manuscript timetables and tracking will influence time-to-acceptance. To the extent that these changes occur during the period of data collection or subsequently, they will influence your model's ability to accurately anticipate time-to-acceptance.

Model fit is checked by examining quantile statistics from the data against the same statistics from the posterior draws. For instance, the percentage of articles in the data with τ less than 90 days is 11, which compares with a percentage of 13.3 from the posterior draws. Continuing, the percentage of articles with τ longer than 360 days from the data is 6, which compares with a percentage of 4.4 from the posterior draws.

The model has practical applications if you wish to meet a deadline. As an example, in order for research to be considered by the 5th Assessment Report of the Intergovernmental Panel on Climate Change (IPCC), you must have your manuscript accepted for publication by March 15, 2013.

The graph in Figure 4.6 shows your model's predictive probability of meeting this deadline versus date. Results are based on 1,000 random draws from a gamma distribution where the scale and shape parameters are derived from 1,000 draws from the joint posterior given the data. The probability is the percentage of posterior τ draws less than τ_o days from March 15, 2013. The 95 percent credible interval shown by

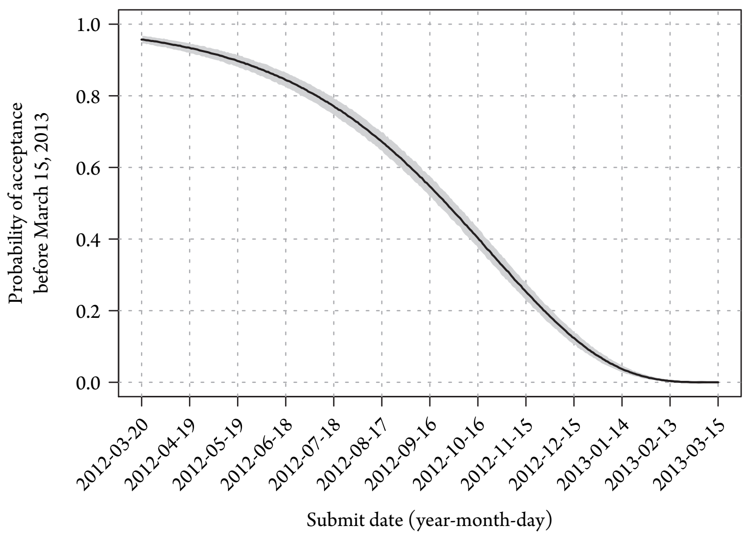

Figure 4.6 Probability of paper acceptance as a function of submit date.

the gray band is obtained by repeating the 1,000 draws from the gamma distribution 1,000 times and taking the 0.025 and 0.975 quantile values of the probabilities. The probability is high in the early months of 2012 when the deadline is still a year in the future. However, by mid-September of 2012, the probability drops below 50 percent and by mid-January 2013, the probability is less than 10 percent.

The predictive probabilities from your model reflect what can be expected if you submit to an arbitrary AMS journal with "hurricane" in the title under the assumption that your paper will be accepted. Certain journals and authors could have a faster or slower turnaround time. In this case, you need to use a hierarchical model (see Chapter 12) to accommodate author and journal differences. With a hierarchical model, you use an MCMC approach to obtain the posterior probabilities.

4.6.2 Markov Chain Monte Carlo approach

MCMC is a class of algorithms for sampling from a probability distribution using a Markov chain. It is a way to obtain samples from your posterior distribution. It is flexible, easy to implement, and requires little input from you. It is one reason behind the recent surge in popularity of Bayesian statistics.

Gibbs sampling is an example of an MCMC algorithm. Suppose your parameter vector of interest is $\theta = (\theta_1, \theta_2, \ldots, \theta_p)$. The joint posterior distribution of theta, which is denoted $[\theta|$ data$]$ may be of high dimension and difficult to summarize. Instead suppose you define the set of conditional distributions as

$$\left[\theta_1|\theta_2, \ldots, \theta_p, \text{data}\right], \left[\theta_2|\theta_1, \theta_3 \ldots, \theta_p, \text{data}\right], \cdots \left[\theta_p|\theta_1, \ldots, \theta_{p-1}, \text{data}\right] \quad (4.9)$$

where $[X|Y,Z]$ represents the distribution of X conditional on values of random variables Y and Z. The idea is that you can set up a Markov chain from the joint posterior distribution by simulating parameters from the set of p conditional distributions.

Drawing one value for each parameter from these distributions in turn is called one update (iteration) of the Gibbs sampling. Under general conditions, draws will converge to the target distribution (joint posterior of θ). Unfortunately, this theoretical result provides no practical guidance on how to decide if the simulated sample provides a reasonable approximation to the posterior density (Jackman, 2009).

4.6.3 JAGS

A popular general purpose MCMC software that implements Gibbs sampling is WinBUGS (Windows version of Bayesian inference Using Gibbs Sampling) (Lunn et al, 2000). It is stand-alone software with a GUI. JAGS (Just Another Gibbs Sampler) is an open-source project written in C++. It runs on any computing platform and can be called from R with functions from the **rjags** package Plummer (2011).

Here you use JAGS on the earlier problem of making inferences about the proportion of U.S. landfalls. The example is simple enough and an MCMC algorithm is not really necessary, but it is instructive for you to see the work flow as you will see it again in Chapter 12. To begin, download and install the latest version of JAGS. This is C++ code that gets installed outside of R. Next open a text file and write the model code in a language that JAGS understands. You can copy and paste from here. Call the file **JAGSmodel.txt**.

```
___JAGS code___
model {
  h ~ dbin(pi, n)  #data
  pi ~ dbeta(a, b) #prior
  a <- 3.26
  b <- 7.19
  }
_______________
```

Note the similarity to R syntax.

Every JAGS (and BUGS) program begins with a model statement followed by an open curly brace and ends with an closed curly brace. The second line specifies the landfall count `h` using the "twiddle" or "tilde" character, which indicates a stochastic relation. The stochastic relation is of the form `node~ddens(arguments)`, where `node` is your parameter and `ddens` is the name of a function calling one of the statistical distributions supported in JAGS. These functions start with the letter `d`, as here in `dbin` for the binomial mass function and `dbeta` for the beta density used in third line for your prior. Here the values of the parameters `a` and `b` are the same as you used earlier in the chapter and are set using the deterministic operator ($<-$).

JAGS and BUGS are declarative languages. R is a procedural language. The JAGS syntax specifies the model, but it does not define the computational steps. When the

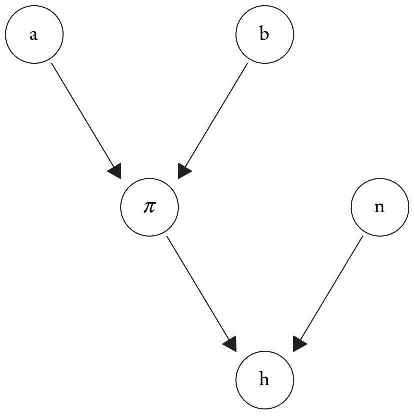

Figure 4.7 Directed acyclic graph of the landfall proportion model.

JAGS model is compiled, the model syntax is turned into a set of sequential instructions, but this is hidden. Because of this, the statement order is not important. What matters is that the compiler can resolve the names and links in the model.

Think of the JAGS model as a directed acyclic graph (DAG) as shown in Figure 4.7. Nodes are the parameters and data and arrows indicate the conditional dependency, either stochastic or deterministic. Hurricane landfall h depends on the number of hurricanes n and the proportion π, where the proportion π depends on your prior assumptions encoded in the beta parameters a and b. Landfall proportion π is conditionally dependent on the scale (a) and shape (b) values through the beta density. The number of landfalls h is dependent on the basin-wide count n and the proportion of landfalls π through the binomial distribution. The conditional independence structure of your model is clear in the DAG. Note that BUGS allows you to generate model code from a DAG.

Next open R and type

```
> require(rjags)
```

You call JAGS from R with the `jags.model` function by typing,

```
> model = jags.model('JAGSmodel.txt',
+    data = list('h'=4, 'n'=34),
+    inits = list('pi'=.5, .RNG.seed=3042,
+    .RNG.name="base::Super-Duper"))
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 7

Initializing model
```

The function sends the model in file **JAGSmodel.txt** to JAGS for parsing and compiling. Make sure the file is in your working directory. The `data` argument specifies the number of landfalling hurricanes *h* and the number of hurricanes *n* as a list. The `inits` argument directs JAGS to initialize the MCMC although that is not strictly necessary with this simple model. The last two elements in the `inits` list specify the random number generator (RNG) and an initial seed for the generator. This allows the results to be replicated as the sequence of random numbers will be identical if the same seed value is used. The object `model` is of class `jags`.

By default, the function generates 1,000 samples (updates) of the MCMC algorithm. This is enough for successive values of π to depart from the initial value and reach the posterior density. In fact, this "burn-in" is instantaneous with this model. If more iterations are needed, you specify the number in the function with the argument `n.adapt` or you can use the `update` method on the model object by typing,

```
> update(model, 1000)
```

An additional 1,000 iterations from the MCMC are generated but not saved.

Finally you use the `coda.samples` function to generate posterior samples of π and save them. It continues updating the chain for the number of iterations specified by `n.iter`, but this time it saves them if the parameter is listed in the `variable.names` argument.

```
> out = coda.samples(model, variable.names='pi',
+    n.iter=1000)
```

CODA stands for COnvergence Diagnostic and Analysis. It describes a suite of functions for analyzing outputs generated from BUGS software.

The object returned is of class `mcmc.list`. You use the `summary` method on this object to obtain a summary of the posterior distribution for π.

```
> summary(out)
Iterations = 1001:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

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

          Mean             SD       Naive SE
       0.16294        0.05342        0.00169
Time-series SE
       0.00134
```

```
2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.0739 0.1228 0.1595 0.1957 0.2878
```

The output shows first the characteristics of your MCMC sampler and then provides statistics on the samples from the posterior distribution. The posterior mean indicating the proportion of all North Atlantic hurricanes that make landfall in the United States is 0.16. Quantiles of π from the posterior samples indicate that the 95 percent credible interval for the proportion is (0.07, 0.29).

You obtain other statistics from the posterior samples by extracting the array corresponding to π from the MCMC list and performing the appropriate computations. For instance, given your prior and your data, how likely is it that the population percentage of landfalls is less than or equal to 0.25? This is answered by typing,

```
> pi = as.array(out)
> sum(pi <= .25)/length(pi)
[1] 0.941
```

Here, because there is only one chain and one variable, you extract the vector from the array using `as.array`. The answer (94 percent) compares well with what you obtained in §4.2 where you used the conjugate model.

Additional analysis can be done with the posterior samples, and the plot method will produce a trace and density plot of your π values by typing

```
> plot(out)
```

The trace plot shows your samples versus the simulation index. The trace plot is useful in assessing whether your chain has converged. Although the samples vary, the mean and variance are relatively constant indicating convergence. This is expected with a simple model. The density plot indicates the distribution of all the samples.

The MCMC approach is flexible and it allows you to easily answer other related inferential questions. For instance, suppose you want to know the probability that any two consecutive hurricanes that form in the North Atlantic will hit the United States. You can add a node to your model of the form `pi2 <- pow(pi,2)`. This is changed in your file **JAGSmodel.txt**. You then monitor this node and draw inferences from the posterior samples. The median posterior probability indicates only a 16 percent chance of consecutive hurricanes hitting the United States. The result assumes that consecutive hurricanes are independent.

4.6.4 WinBUGS

WinBUGS and OpenBUGS have functions for modeling spatial data that are not yet available in JAGS. In Chapter 12, you will build a Bayesian space-time model for your hurricane data. Here we show you the work flow for implementing a Bayesian model in WinBUGS through R. If you are using Windows, there is no problem. If you are using a Mac, your options are limited. One is to run WinBUGS with Wine. Another is to use OpenBUGS without using R.

Here we assume that you are using a Windows machine or running Windows on a partition of your Mac (e.g., bootcamp). First download and install WinBUGS?[2] Make sure you have permission to write files in the directory where the executable file is stored. Next, write a WinBUGS model describing the time-to-acceptance for an article on hurricanes submitted to the AMS (see §4.6.1). The code is given as follows.

```
___WinBUGS code___
model {
  for (i in 1:N) {
    TTA[i] ~ dgamma(shape, rate)
    }
  ttastar ~ dgamma(shape, rate)
  ht <- step(120-ttastar)
    shape ~ dgamma(.1, .1)
    rate ~ dgamma(.1, .1)
}
_______________
```

The syntax is nearly identical to JAGS, and in this case you can run this as a JAGS model using the work flow described previously. Save the code in **WinBUGSmodel.txt** outside of R. The `step` function is 1 if its argument is positive and 0 otherwise. The parameter `ht` thus indicates whether the sampled time-to-acceptance is less than 120 days.

Next prepare the data inputs required by the `bugs` function from the **R2WinBUGS** package. This is a list containing the name of each data vector.

```
> require(R2WinBUGS)
> N = length(tau)
> TTA = as.numeric(tau)
> data = list("TTA","N")
```

Using these data together with your model, you run an MCMC sampler to get estimates for `ttastar`. Beforehand you decide how many chains to run for how many iterations. If the length of the burn-in is not specified, then the burn-in is taken as half the number of iterations. You also specify the starting values for the chains. Here you do this by writing the following function

```
> inits=function(){
+   list(shape=runif(1, .3, .5), rate=runif(1, .3, .5),
+     ttastar = runif(1, 100, 120))
+ }
```

[2] `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml`

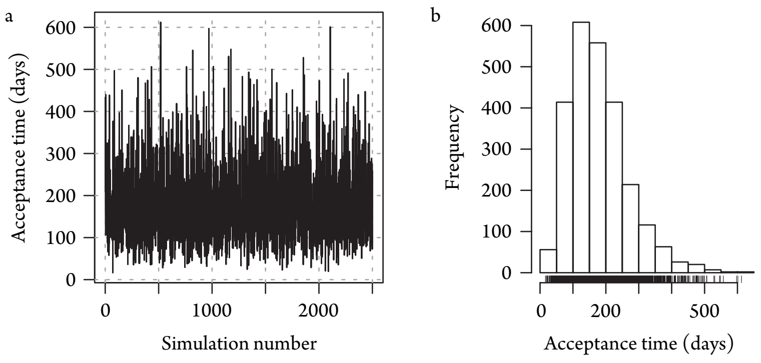

Figure 4.8 Posterior samples of time-to-acceptance. (a) Trace plot and (b) histogram.

You then start the MCMC by typing

```
> model = bugs(data, inits,
+    model.file="WinBUGSmodel.txt",
+    parameters=c("ttastar", "ht", "shape", "rate"),
+    n.chains=3, n.iter=5000, n.thin=1,
+    bugs.directory="C:/Program Files/WinBUGS14")
```

The argument `bugs.directory` must point to the directory containing the WinBUGS executable, and you must have write permission in this directory.

The results are saved in the object `model`. The `bugs` function uses the parameter names you gave it in the WinBUGS text file to structure the output into scalar, vector, and arrays containing the samples. The object `model$sims.array`, as example, contains an array of the posterior samples. In addition, the samples are stored in a vector. You use the `print` method to get a summary of your results. Additionally, you use the `plot` method to create graphs summarizing the posterior statistics of your parameters (see Figure 4.8) and to display diagnostics relating to chain convergence. The chains are convergent when your samples are taken from the posterior.

By default, `bugs` removes the first half of the iterations as "burn-in", where the samples are moving away from the initial set of values toward the posterior distribution. Samples can be thinned if successive values are highly correlated (see Chapter 12). The burn-in and thinning determine the final number of samples (saved in `model$n.keep`) available for inference.

Here you are interested in the posterior samples of the time-to-acceptance `ttastar`. To plot the values as a sequence (trace plot) and as a distribution, type

```
> plot(model$sims.array[, 1, 1], type="l")
> plot(density(model$sims.array[, 1, 1]))
```

The trace plot shows the sequence of sample values by the simulation number. The mean and variance of the values do not appear to change across the sequence indicating the samples are coming from the posterior distribution. A view of the marginal distribution of time-to-acceptance is shown with a histogram.

Because your model contains a node indicating whether the sample time-to-acceptance is greater than 120 days, you determine the posterior probability of this occurrence by typing

```
> sum(model$sims.array[, 1, 2])/model$n.keep
[1] 0.279
```

The answer of 27.9 percent is close to what you obtained in §4.6.1.

Greater flexibility in summarizing and plotting your results are available with functions in the **coda** package (Plummer et al., 2010). You turn on the `codaPkg` switch and rerun the simulations.

```
> modelc = bugs(data, inits,
+    model.file="WinBUGSmodel.txt",
+    parameters=c("ttastar", "ht", "shape", "rate"),
+    n.chains=3, n.iter=5000, n.thin=1, codaPkg=TRUE,
+    bugs.directory="C:/Program Files/WinBUGS14")
```

The saved object `modelc` is a character vector of file names with each file containing coda output for one of the three chains. You create an MCMC list object (see §4.6.3) with the `read.bugs` function.

```
> out = read.bugs(modelc, quiet=TRUE)
```

Additional plotting options are available using the **lattice** package. To plot the density of all the model parameters, type

```
> require(lattice)
> densityplot(out)
```

The plots contain distributions of the MCMC samples from all the model nodes and for the three chains. Density overlap across the chains is another indication that the samples are from the posterior.

This chapter introduced Bayesian statistics for hurricane climate. The focus was on the proportion of landfalling hurricanes. We examined a conjugate model for this proportion. We also looked at the time-to-acceptance for a manuscript. The conjugate model revealed the mechanics of the Bayesian approach. We also introduced MCMC samplers. This innovation allows you flexibility in creating models for your data. The next chapter shows you how to graph and plot your data.