12

BAYESIAN MODELS

"Errors using inadequate data are much less than those using no data at all."
—Charles Babbage

In this chapter, we focus on Bayesian modeling. Information about past hurricanes is available from instruments and written accounts. Written accounts are generally less precise than instrumental observations, which tend to become even more precise as technology advances. Here we show you how to build Bayesian models that make use of the available information while accounting for differences in levels of precision. We begin with a model for U.S. hurricane frequency and finish with a space–time model for basin-wide occurrences.

12.1 LONG-RANGE OUTLOOK

We start with a model for predicting U.S. hurricane activity over the next three decades. The model is useful as a benchmark for climate change studies. The methodology was originally presented in Elsner and Bossak (2001) based on the formalism given by Epstein (1985).

12.1.1 Poisson-Gamma Conjugate

As you have seen throughout this book, the arrival of hurricanes on the coast is usefully considered a stochastic process, where the annual counts are described reasonably well by a Poisson distribution. The Poisson distribution is a limiting form of the binomial distribution with no upper bound on the number of occurrences and where the parameter λ characterizes the rate process. Knowledge of λ allows you to make statements about future hurricane frequency. Since the process is stochastic, your statements will be given in terms of probabilities (see Chapter 7).

Elsner: "12˙ELSNER˙CH12" — 2012/9/24 — 19:11 — page 301 — #1

For example, the probability of $\hat{h}$ hurricanes occurring over the next T years (e.g., 1, 5, 20, etc.) is

$$f(\hat{h}|\lambda, T) = \exp(-\lambda T)\frac{(\lambda T)^h}{h!} \quad \text{for} \quad h = 0, 1, \ldots, \quad \lambda > 0, \quad \text{and} \quad T > 0 \tag{12.1}$$

The hat notation is used to indicate future values.

The parameter λ and statistic T appear in the formula as a product, which is the mean and variance of the distribution. Knowledge about λ can come from historical written archives and instrumental records. It is logical for you to want to use as much of this information as possible before inferring something about future activity.

This requires you to treat λ as a parameter that can be any positive real number, rather than as a fixed constant. One form for expressing your judgment about the values λ can take is through the gamma distribution. The numbers that are used to estimate λ from a set of data are the time interval T' and the number of hurricanes h' that occurred during this interval.[1] For instance, observations from the hurricane record since 1851 indicate 15 hurricanes over the first 10 years, so $T' = 10$ and $h' = 15$. To verify this, type

```
> H = read.table("US.txt", header=TRUE)
> sum(H$All[1:10])
[1] 15
```

The gamma distribution of possible future values for λ is given by

$$f(\hat{\lambda}|h', T') = \frac{T'^{h'}\lambda^{h'-1}}{\Gamma(h')}\exp(-\lambda T') \tag{12.2}$$

with the expected value $\mathrm{E}(\hat{\lambda}) = h'/T'$, and the gamma function $\Gamma(x)$ given by

$$\Gamma(x) = \int_0^\infty t^{x-1}\mathrm{e}^{-t}\mathrm{d}t \tag{12.3}$$

Of importance here is the fact that if the probability density on $\hat{\lambda}$ is a gamma distribution, with initial numbers (prior parameters) h' and T', and the numbers h and T are later observed, then the posterior density of $\hat{\lambda}$ is also gamma with parameters $h + h'$ and $T + T'$. In other words, the gamma density is the conjugate prior for the Poisson rate λ (Chapter 4).

12.1.2 Prior Parameters

The above formulation gives you a convenient way to combine earlier less reliable information with later information. You simply add the prior parameters h' and T' to the sample numbers h and T to get the posterior parameters.

But how do you estimate the prior parameters? You have data but the values could be too low (or too high) due to missing or misclassified hurricanes. One way is to

[1] The prime notation indicates prior (here, earlier) information.

use bootstrapping. Bootstrapping is sampling from your sample (resampling) to provide an estimate of the variation about your statistic of interest (see Chapter 3). Here you use the `bootstrap` function in the **bootstrap** package (Tibshirani and Leisch, 2007) to obtain a confidence interval about λ from data before 1899.

First load the package and save a vector of the counts over the earlier period of record. Then to get a bootstrap sample of the mean, use the `bootstrap` function on this vector of counts.

```
> require(bootstrap)
> early = H$All[H$Year < 1899]
> bs = bootstrap(early, theta=mean, nboot=1000)
```

To obtain a 90 percent bootstrapped confidence interval about the mean, type

```
> qbs = quantile(bs$thetastar, prob=c(.05, .95))
> qbs
  5%  95%
1.42 2.10
```

Although you cannot say with certainty what the true hurricane rate was over this early period, you can make a sound judgment that you are 90 percent confident that the interval contains it. In other words you are willing to admit a 5 percent chance that the true rate is less than 1.42 and a 5 percent chance that it is greater than 2.1.

Given this appraisal of your belief about the early hurricane, rate you need to obtain an estimate of the parameters of the gamma distribution. Said another way, given your 90 percent confidence interval for the rate, what is your best estimate for the number of hurricanes and the length of time over which those hurricanes occurred?

You do this with the optimization function `optim`. You start by creating an objective function defined as the absolute value of the difference between gamma quantiles and your target quantiles.

```
> obj = function(x){
+   sum(abs(pgamma(q=qbs, shape=x[1], rate=x[2]) -
+   c(.05, .95)))
+ }
```

You then apply the optimization function to your objective function starting with reasonable initial values for the gamma parameters given in the `par` argument and save the solution in the vector `theta`.

```
> theta = optim(par = c(2, 1), obj)$par
```

Store these parameters as separate objects by typing

```
> hp = theta[1]
> Tp = theta[2]
```

This procedure quantifies your judgment about hurricanes before the reliable set of counts. It does so in terms of the shape and rate parameter of the gamma distribution.

12.1.3 Posterior Density

You now have two distinct pieces of information from which to obtain a posterior distribution for λ (landfall rate). Your prior parameters $h' = 69.7$ and $T' = 39.9$ from above and your likelihood statistics based on the data over the reliable period of record (1899–2010). The total number of hurricanes over this reliable period and the record length are

```
> late = H$All[H$Year >= 1899]
> h = sum(late)
> T = length(late)
> h; T
[1] 187
[1] 112
```

The posterior parameters are therefore $h'' = h + h' = 256.7$ and $T'' = T + T' = 151.9$. Note that although the likelihood parameters h and T must be integers, the prior parameters can take on any real value depending on your degree of belief.

Since the prior, likelihood, and posterior are in the same gamma family, you can use `dgamma` to compute the densities.

```
> curve(dgamma(x, shape=h + hp, rate=T + Tp), from=1,
+   to=3, xlab="Landfall Rate [hur/yr]",
+   ylab="Density", col=1, lwd=4, las=1)
> curve(dgamma(x, shape=h, rate=T), add=TRUE,
+   col=2, lwd=4)
> curve(dgamma(x, shape=hp, rate=Tp), add=TRUE,
+   col=3, lwd=4)
> legend("topright", c("Prior", "Likelihood",
+   "Posterior"), col=c(3, 2, 1), lwd=c(3, 3, 3))
```

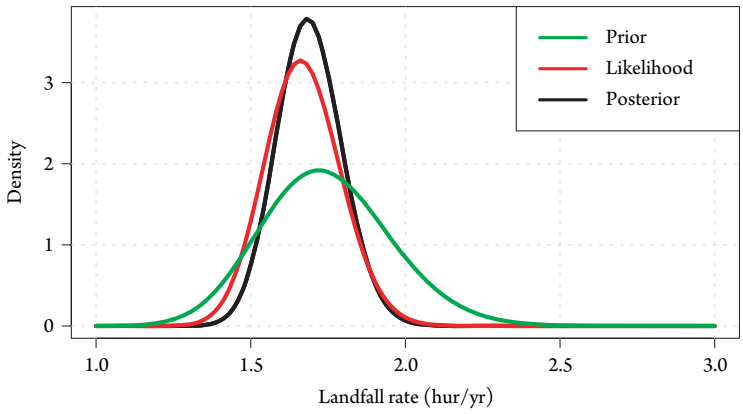

Figure 12.1 Gamma densities for the landfall rate.

The densities are shown in Figure 12.1. Note that the posterior density resembles the likelihood but is shifted in the direction of the prior. It is also narrower. 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. The relatively broad density on the prior estimate indicates low precision. Combining the prior and likelihood results in a posterior distribution that represents your best information about λ.

12.1.4 Predictive Distribution

The information you have about λ is contained in the two parameters h'' and T'' of the gamma density. Of practical interest is how to use this information to predict future hurricane activity. The answer lies in the fact that the *predictive density* for observing $\hat{h}$ hurricanes over the next $\hat{T}$ years is a negative binomial distribution, with parameters h'' and $\frac{T''}{\hat{T}+T''}$ given by

$$f\left(\hat{h}|h'', \frac{T''}{\hat{T}+T''}\right) = \frac{\Gamma(\hat{h}+h'')}{\Gamma(h'')\hat{h}!}\left[\frac{T''}{\hat{T}+T''}\right]^{h''}\left[\frac{\hat{T}}{\hat{T}+T''}\right]^{\hat{h}} \tag{12.4}$$

The mean and variance of the negative binomial are $\hat{T}\frac{h''}{T''}$ and $\hat{T}\frac{h''}{T''}\left(\frac{\hat{T}+T''}{T''}\right)$, respectively. Note that the variance of the predictive distribution is larger than it would be if λ were known precisely. If you are interested in the probability of a hurricane *next* year, then $\hat{T}$ is one and small compared with T'' so it makes little difference, but if you are interested in the distribution of hurricane activity over the next 20 years, then it is important.

You plot the posterior probabilities and cumulative probabilities for the number of U.S. hurricanes over the next 10 years by typing,

```
> Th = 10
> m = Th * (h + hp)/(T + Tp)
> v = Th * m * (Th + T + Tp)/(T + Tp)
> nl = 0:32
> hh = dnbinom(nl, mu=m, size=v)
> par(las=1, mar=c(5, 4, 2, 4))
> plot(nl, hh, type="h",
+   xlab="Number of U.S. Hurricanes",
+   ylab="Probability", col="gray", lwd=4)
> par(new=TRUE)
> p = pnbinom(nl, mu=m, size=v)
> plot(nl, p, type="l", col="red", xaxt="n", yaxt="n",
+    xlab="", ylab="", lwd=2)
> axis(4)
> mtext("Probability of h less than or equal to H",
+    side=4, line=2.5, las=0)
```

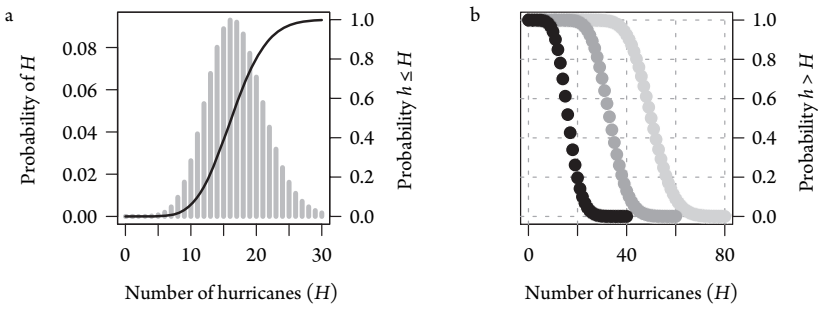

Figure 12.2 Predictive probabilities. (a) 10 years and (b) 10, 20, and 30 years.

Results are shown in Figure 12.2, where the probabilities of H hurricanes in 10 years are displayed with vertical bars and with a scale plotted along the left vertical axis, and the probability that the number of hurricanes will be less than or equal to H is displayed as a solid curve and with a scale along the right axis. The probability that the number of hurricanes will exceed H is shown for a random 10-, 20-, and 30-year period is shown in the right panel. The expected number of U.S. hurricanes over the next 30 years is 51, of which 18 (not shown) are anticipated to be major hurricanes. These probabilities represent the best estimates of the future baseline hurricane climate.

The above approach is a rational and coherent foundation for incorporating all available information about hurricane occurrences, while accounting for the differences in the precision as it varies over the years. It could be used to account for the influence of climate change by discounting the older information. That is, records influenced by recent changes can be given more weight than records from earlier decades.

12.2 SEASONAL MODEL

Here you create a Bayesian model for predicting annual basin-wide hurricane counts. The counts are reliable starting in the middle twentieth century. But data records on past hurricanes extend farther back, and these earlier records are useful to understand and predict seasonal activity. The logarithm of the annual rate is linearly related to sea-surface temperature (SST) and the Southern Oscillation Index (SOI) as discussed in Chapter 7.

Samples from the posterior distribution are generated using the Markov chain Monte Carlo approach discussed in Chapter 4. The JAGS code is given below. You copy and paste the code into a text file in your working directory with the name *JAGSmodel2.txt*.

```
___JAGS code___
model {
for(i in 1:N) {
   h[i] ~ dpois(lambda[i])
   mu[i] <- b0 + b1*SOI[i] + b2*SST[i] + b3*RI[i] +
```

```
            eta[i]
    lambda[i] <- exp(mu[i])
    tt[i] <- tau[RI[i] + 1]
    eta[i] ~ dnorm(0, tt[i])
    }
  b0 ~ dnorm(0, .0001)
  b1 ~ dnorm(0, .0001)
  b2 ~ dnorm(0, .0001)
  b3 <- log(pm)
  pm ~ dunif(lo, hi)
  tau[1] ~ dgamma(.001, .001)
  tau[2] ~ dgamma(.001, .001)
}
```

The code specifies the regression model and associated priors. In particular, the annual hurricane count `h[i]` is stochastically generated from a Poisson distribution with a rate `lambda[i]`. The parameter `lambda[i]` is deterministically linked to `mu[i]` through the exponential function, where `mu[i]` is linearly related to the SST and SOI covariates and a "reliability" index (`RI[i]`). The reliability index is coded as 1 for years prior to aircraft and 0 for years after. Here you use 1943 as the cutoff year.

Your model includes a random effects term (`eta[i]`) that quantifies the extra variation in annual hurricane rates not modeled by the covariates. It accommodates two variance distributions to account for the different levels of uncertainty before and after the cutoff year. JAGS is a declarative language so the order of the assignment statements is irrelevant. Your model is a directed acyclic graph (DAG), where the nodes are the parameters and data and the arrows indicate the conditional dependency, either stochastic or deterministic (see Chapter 4).

Load the annual data and create a data frame that includes only the data to be modeled. Here the basin-wide hurricane count `H`, the August–October value of SOI, the August through October value of SST, and a reliability index `RI` that is coded as 1 for years prior to aircraft and 0 for years after.

```
> load("annual.RData")
> dat = data.frame(Yr=annual$Year, H=annual$B.1,
+   SOI=annual$soi, SST=annual$sst,
+   RI=as.numeric(annual$Year < 1943))
> dat = dat[16:160, ]
```

You execute the model using the `jags.model` function in **rjags** (Plummer, 2011). The function specifies the file name, the data as a list, initial values as a list, as well as the number of chains and the number of initial updates (`n.adapt`).[2]

[2] You can monitor the progress by specifying `options(jags.pb="text")` [or `="gui"`] before your call to `jags.model`.

```
> require(rjags)
> model = jags.model('JAGSmodel2.txt',
+   data=list(N=length(dat$H), h=dat$H,
+   SST=dat$SST, SOI=dat$SOI, RI=dat$RI,
+   lo=.2, hi=.95),
+   inits = list(b0=0, b1=0, b2=0,
+   tau=c(.1, .1), pm=.9, .RNG.seed=3042,
+   .RNG.name="base::Super-Duper"),
+   n.chains = 2,
+   n.adapt = 1000)
```

You continue sampling by applying the `update` function on the model object and specifying the number of additional samples. You then save the last 1,000 samples of the SST (`b1`) and SOI (`b2`) coefficients.

```
> update(model, 2000)
> out = coda.samples(model, c('b1', 'b2'), 1000)
```

The `coda.samples` is a wrapper function[3] for `jags.samples` that monitors your requested nodes, updates the model, and outputs the samples to a single `mcmc.list` object. The first argument is the model object, the second is a vector of variable names, and the third is the number of iterations. To examine the samples, you plot histograms (Fig. 12.3).

```
> par(mfrow=c(1, 2))
> hist(out[[1]][, 1], xlab=expression(beta[1]))
> hist(out[[1]][, 2], xlab=expression(beta[2]))
```

The chain number is given inside the double brackets and the sample values are given as a matrix where the rows are the consecutive samples and the columns are the

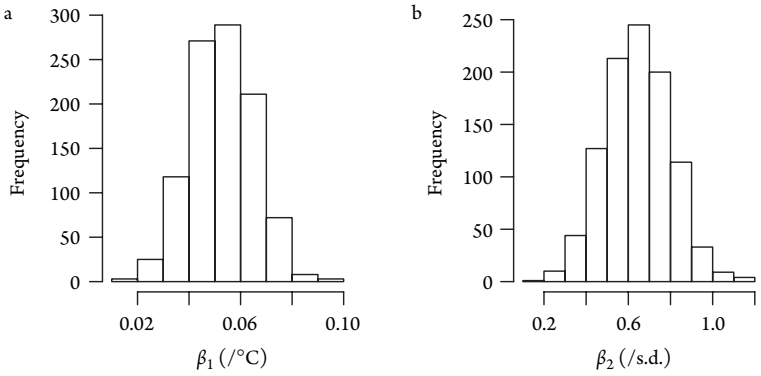

Figure 12.3 Posterior samples for the (a) SST (β_1) and (b) SOI (β_2) parameters.

[3] A function whose main purpose is to call another function.

parameters. The distributions are shifted to the right of zero verifying that both the SST and the SOI are important in modulating annual hurricane rates across the North Atlantic.

A time series of box plots shows the distribution of the annual rate parameter (λ) as a function of year. First, generate additional samples from the posterior, this time monitoring `lambda`.

```
> out = coda.samples(model, c('lambda'), 1000)
```

Then use the `fivenum` function with the `points` and `lines` functions within a loop over all years.

```
> plot(c(1866, 2010), c(0, 20), type="n", bty="n",
+   xlab="Year", ylab="Annual Rate (hur/yr)")
> for(i in 1:dim(dat)[1]){
+ points(dat$Yr[i], fivenum(out[[1]][, i])[3], pch=16)
+ lines(c(dat$Yr[i], dat$Yr[i]),
+   c(fivenum(out[[1]][, i])[1],
+   fivenum(out[[1]][, i])[5]))
+ }
```

The resulting plot is shown in Figure 12.4. The model is useful for describing the annual rate variation and the associated uncertainty levels. The median rate is given as a point and the range is given as a vertical line. Hurricane rates appear to fluctuate about the value of 5 hur/yr until about 1945 when they appear to increase slightly before falling back again during the 1970s and 1980s. There is a more substantial increase beginning in the middle 1990s.

The extra variation in the rate specific to each year that is not modeled by the two covariates is quantified with the term `eta`. The variation includes the two levels of data precision before and after the start of the aircraft reconnaissance era. A time series plot of posterior samples of `eta` are shown in Figure 12.5. The posterior median is plotted as a point and the vertical lines extend from the 25th to the 75th percentiles. The red line is a local regression smoother through the median values. As expected, the

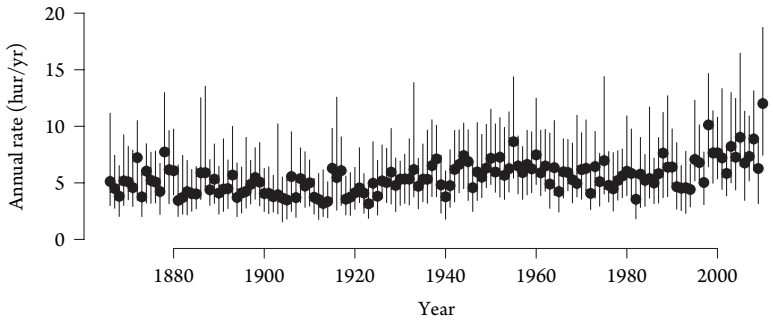

Figure 12.4 Modeled annual hurricane rates.

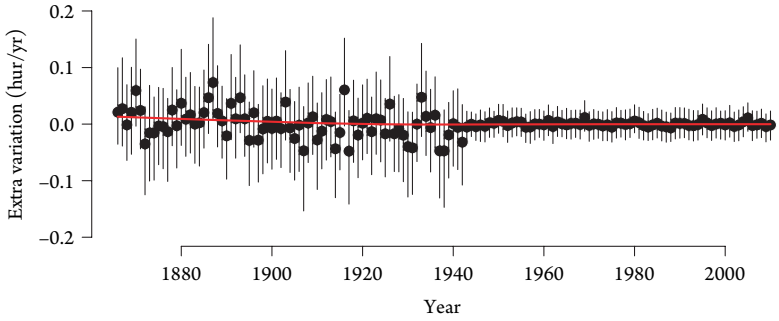

Figure 12.5 Extra variation in hurricane rates.

graph shows larger unexplained variations and a tendency for underprediction of the rates (positive values) during the earlier years, but much less so afterward.

12.3 CONSENSUS MODEL

In choosing one model over another you typically apply a selection procedure on a set of covariates to find the single "best" model. You then make predictions with the model as if it had generated the data. Unfortunately, this approach ignores the uncertainty in your model selection procedure resulting in over confidence in your predictions.

For example, given a pool of covariates for hurricane activity, a stepwise regression procedure is typically employed to search through hundreds of candidate models (combinations of covariates). The model that provides the best level of skill is subsequently chosen. The best model is then subjected to a leave-one-out cross-validation (LOOCV) exercise to obtain an estimate of how well it will predict future data. This procedure unfortunately does not result in a true cross-validation as the process of selecting the covariates is itself not cross-validated. A true cross-validation assesses how well an algorithm for choosing a particular model (including the predictor selection phase) will do in forecasting the unknown future (see Chapter 7).

An alternative to choosing a single "best" model is to use Bayesian model averaging (BMA). BMA works by assigning a probability to each model (combination of covariates), then averaging over all models weighted by their probability (Raftery et al., 2005). Here you produce a consensus forecast of seasonal hurricane activity. In doing so you we show how the approach can facilitate a physical interpretation of your modeled relationships. The presentation follows closely the work of Jagger and Elsner (2010).

12.3.1 Bayesian Model Averaging

Let H_i, $i = 1,\ldots,N$ denote your set of observed hurricane counts by year. Assume that your model has k covariates, then let $\mathbf{X}$ be the covariate matrix with components

$X[i,j+1], i = 1,\ldots,N, j = 1,\ldots k$ associated with the ith observation of the jth covariate and with the intercept term $X[i,1] = 1$ for all i. Associated with the intercept and k covariates are $k+1$ parameters $\beta_j, j = 1,\ldots,k+1$.

You assume that the counts are adequately described by a Poisson distribution. The logarithm of the rate is regressed onto the covariates as

$$H_i \sim \text{pois}(\lambda_i)$$
$$\log(\lambda_i) = \sum_{j=1}^{k+1} X[i,j]\beta_j$$

This is a generalized linear model (GLM) and the method of maximum likelihoods is used to estimate the parameters (Chapter 7). From these parameter estimates and the values of the corresponding covariates, you infer λ from the regression equation. The future hurricane count conditional on these covariates is described by a Poisson distribution with a mean of λ. Thus your model is probabilistic, and the average count represents a single forecast.

A full model is defined as one that uses all k covariates. However, it is usual that some of the covariates do not contribute much to the model. In frequentist statistics, these are the ones that are not statistically significant. You can choose a reduced model by setting some of the k parameters to zero. Thus, with k covariates, there are a total of $m = 2^k$ possible models. The idea behind BMA is that all the m models are used with a probability assigned to each. Predictions are made by a weighted average over the predictions made with each model and where the weights are the model probabilities. Models with greater probability carry proportionally more weight in the average.

Consider a simple case. You have observations of Y arising from either one of two possible regression models. Let $Y_1 = \alpha_1 + \epsilon_1$ be a constant mean model and $Y_2 = \alpha_2 + \beta x + \epsilon_2$ be a simple regression model where x is a single covariate. The residual terms ϵ_1, ϵ_2 are independent and normally distributed with means of zero and variances of σ_1^2 and σ_2^2, respectively.

Suppose that you assign a probability p that the constant mean model generated the observed data. Then there is a probability $1-p$ that the simple regression model generated the data instead. With BMA, the posterior predictive expectation (mean) of Y is $\mu = p\mu_1 + (1-p)\mu_2 = p\alpha_1 + (1-p)(\alpha_2 + \beta x)$. This represents a consensus opinion that combines information from both models as opposed to choosing one over the other.

The posterior predictive distribution of Y given the data is not necessarily normal. Instead it is a mixture of normal distributions with a posterior predicted variance of $p\sigma_1^2 + (1-p)\sigma_2^2 + p(1-p)(\alpha_2 + \beta x - \alpha_1)^2$. This variance under BMA is larger than a simple weighted sum of the individual model variances by an amount $p(1-p)(\alpha_2 + \beta x - \alpha_1)^2$ that represents the uncertainty associated with model choice. Thus, the predictive distribution under BMA has a larger variance than the predictive distribution of a single model.

Over a set of competing models, you need a way to assign a probability to each. You start with a collection of models $M_i, i = 1,\ldots,m$, where each is a unique description of

your data. For example, in the above example, you assign a probability to the constant mean model and a probability to the simple regression model under the constraint that the total probability over both is one.

Now with your data D and a set of proposed models M_i, you determine the probability of your data given each model $[P(D|M_i)]$. You also assign a prior probability to each $[P(M_i)]$ representing your belief that the model generated your data. Under the situation in which you are maximally noncommittal on a model before hand you assigned $1/m$ to each model's prior probability. For example, in the above case, if you believe both models are equally likely, then you assign $P(M_1) = P(M_2) = 0.5$. Using Bayes rule (Chapter 4), you find the probability of the model given, the data as $P(M_i|D) = P(D|M_i) \times P(M_i)/P(D)$ since $P(D)$ is fixed for all models you can let $W_i = P(D|M_i) \times P(M_i)$ be the model weights with probabilities $P(M_i|D) = W_i/\sum_{i=1}^{m} W_i$.

Let the random variable H represent the prediction of a future hurricane count. The posterior distribution of H at h under each model is given by $f(h|D,M_i)$. The marginal posterior probability over all models is given by

$$f(h|D) = \sum_{i=1}^{m} f(h|D,M_i)P(M_i|D) \tag{12.5}$$

A point estimate for the future count (i.e., the posterior mean of H over the models) is obtained by taking the expectation of H given the data as

$$\mathbb{E}(H|D) = \sum_{h=0}^{\infty} hf(h|D) \tag{12.6}$$

Expanding $f(h|D)$ and switching the order of summation, you get

$$\mathbb{E}(H|D) = \sum_{i=1}^{m} P(M_i|D) \sum_{h=0}^{\infty} hf(h|D,M_i) \tag{12.7}$$

which is

$$\sum_{i=1}^{m} P(M_i|D)\mathbb{E}(H|D,M_i) \tag{12.8}$$

where $\mathbb{E}(H|D,M_i) = \mu_i$. For a given model, $P(D|M_i)$ is the marginal likelihood over the parameter space. In other words, $P(D|M_i) = \int P(D|M_i,\theta)f(\theta|M_i)d\theta$, where $f(\theta|M_i)$ is the prior distribution of the parameters for model M_i and $P(D|M_i,\theta)$ is the likelihood of the data given the model $[L(\theta;M_i,D)]$.

The above integral cannot always be evaluated analytically or it may be infinite as when an improper prior is put on the parameter vector θ. In these cases, approximation methods can be used (Hoeting et al., 1999). Here you use the Bayesian Information Criterion (BIC) approximation, which is based on a Laplace expansion of the integral about the maximum likelihood estimates (Madigan and Raftery, 1994).

In summary BMA keeps all candidate models assigning a probability based on how likely it would be for your data to have come from each model. A consensus model, representing a weighted average of all models, is then used to make predictions. If values for the prior parameters come from reasonably well-behaved distributions, then a consensus model from a BMA procedure yields the lowest mean square error of any single best model (Raftery and Zheng, 2003).

BMA provides better coverage probabilities on the predictions than any single model (Raftery and Zheng, 2003). Consider a data record split into a training and testing set. Using the training set, you can create $1 - \alpha$ credible intervals on the predictions. Then, using the testing set, you can calculate the proportion of observations that lie within the credible intervals. This is called the coverage probability. In standard practice with a single best model, the credible intervals are too small resulting in coverage probabilities less than $1 - \alpha$. Since BMA provides a larger variance than any model individually, the coverage probabilities on the predictions are greater or equal to $1 - \alpha$.

12.3.2 Data Plots

You use the data saved in file *annual.RData* and described in Chapter 6. Load the data and create a new data frame that is a subset of the data for years since 1866.

```
> load("annual.RData")
> dat = annual[annual$Year >= 1866, ]
```

The counts are the number of near-coastal hurricanes passing through the regions shown in Figure 6.2. You consider monthly values of SST, SOI, NAO, and sunspots as covariates.

The monthly covariate values are shown in Figure 12.6 as image plots. The monthly values for May through October displayed on the vertical axis are plotted as a function of year displayed on the horizontal axis. The values are shown using a color ramp from blue (low) to yellow (high). The SST and sunspot number (SSN) covariates are characterized by high month-to-month correlation as can be seen by the vertical striations.

12.3.3 Model Selection

You assume that the logarithm of the annual hurricane rate is a linear combination of a fixed subset of your covariates (Poisson generalized linear model). With six months and four environmental variables per month, you have 2^{24} or more than 16.7 million possible models.

Model selection is done with functions in the **BMA** package (Raftery et al., 2009). Obtain the package and source additional functions in **bic.glm.R** that allows you to make posterior predictions.

```
> require(BMA)
> source("bic.glm.R")
```

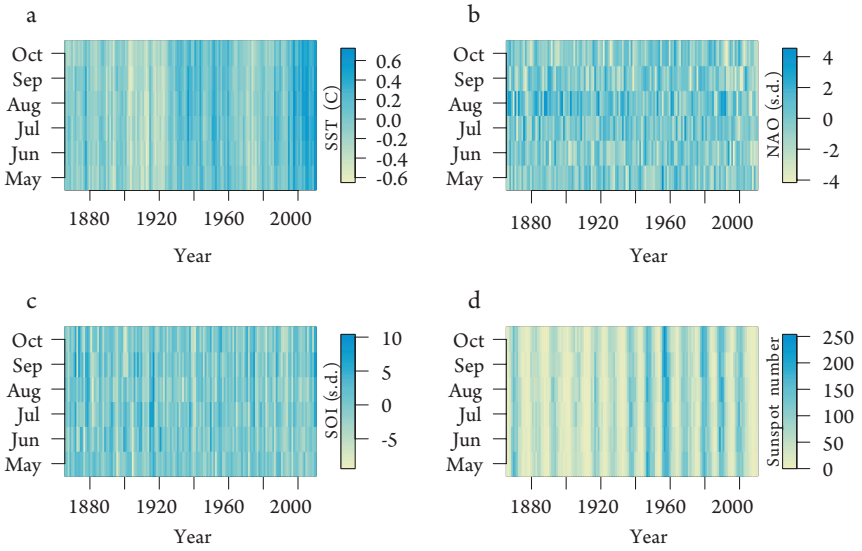

Figure 12.6 Covariates by month. (a) SST, (b) NAO, (c) SOI, and (d) sunspot number.

Specifically, you use the `bic.glm` function for determining the probability on each model and the `imageplot.bma` function for displaying the results. First save the model formula involving the response variable (here `US.1`) and all covariates.

```
> fml = US.1 ~ sst.Oct + sst.Sep + sst.Aug + sst.Jul +
+   sst.Jun + sst.May + nao.Oct + nao.Sep + nao.Aug +
+   nao.Jul + nao.Jun + nao.May + soi.Oct + soi.Sep +
+   soi.Aug + soi.Jul + soi.Jun + soi.May + ssn.Oct +
+   ssn.Sep + ssn.Aug + ssn.Jul + ssn.Jun + ssn.May
```

Then save the output from the `bic.glm` function by typing

```
> mdls = bic.glm(f=fml, data=dat, glm.family="poisson")
```

The function returns an object of class `bic.glm`. By default, only models with a BIC within a factor of 20 of the model with the lowest BIC are kept, and only the top 150 models of each size (number of covariates) are considered. The summary method is used to display the results. Information on the top three models having the lowest BIC values (highest posterior probabilities) is shown in Table 12.1.

```
> summary(mdls, n.models=3, digits=2)
```

The first column lists the covariates (and intercept). The second column gives the posterior probability that a given coefficient is not zero over all the 209 models. One could view this as the inclusion probability. That is, what is the probability that the associated covariate had a nonzero coefficient in the data generating model? For example, the posterior probability that the June NAO covariate is in the data

Table 12.1 Selected models from a BMA procedure.

	$p! = 0$	*EV*	*SD*	*Model* 1	*Model* 2	*Model* 3
Intercept	100	7.5e-01	0.10695	0.692	0.695	0.698
sst.Oct	3.2	7.5e-03	0.07167	.	.	.
sst.Sep	5.8	−3.3e-02	0.21834	.	.	.
sst.Aug	5.0	−3.4e-03	0.15889	.	.	.
sst.Jul	23.9	1.7e-01	0.39788	.	.	.
sst.Jun	5.5	1.5e-02	0.11962	.	.	.
sst.May	3.4	−5.9e-03	0.11293	.	.	.
nao.Oct	2.2	6.1e-04	0.00729	.	.	.
nao.Sep	3.2	1.2e-03	0.00973	.	.	.
nao.Aug	1.9	3.9e-04	0.00564	.	.	.
nao.Jul	3.4	1.6e-03	0.01198	.	.	.
nao.Jun	43.3	−4.3e-02	0.05662	−0.105	−0.096	−0.103
nao.May	5.4	−3.2e-03	0.01713	.	.	.
soi.Oct	37.7	2.0e-02	0.02841	0.055	.	.
soi.Sep	4.6	1.5e-03	0.00869	.	.	.
soi.Aug	25.0	1.3e-02	0.02519	.	.	0.054
soi.Jul	39.2	2.4e-02	0.03381	.	0.054	.
soi.Jun	9.1	−4.6e-03	0.01740	.	.	.
soi.May	1.4	2.0e-04	0.00402	.	.	.
ssn.Oct	7.0	−3.8e-04	0.00178	.	.	.
ssn.Sep	94.4	−1.1e-02	0.00442	−0.012	−0.012	−0.012
ssn.Aug	1.1	1.1e-05	0.00041	.	.	.
ssn.Jul	1.7	−3.7e-05	0.00055	.	.	.
ssn.Jun	88.2	8.5e-03	0.00440	0.010	0.011	0.011
ssn.May	5.7	3.0e-04	0.00149	.	.	.
nVar				4	4	4
BIC				171.072	171.511	171.528
post prob				0.040	0.032	0.031

generating model is 43.3 percent. The third and fourth columns are the posterior expected value (EV) and standard deviation (SD) across all models. Subsequent columns include the most probable models as indicated by values in rows corresponding to a covariate. The number of variables in the model, the model BIC, and the posterior probability are also given in the table.

Models are ordered by BIC values with the first model having the lowest BIC the second model having the second lowest BIC, and so on. The BIC value for a given model is

$$-2 \cdot \ln L + k \ln(n), \tag{12.9}$$

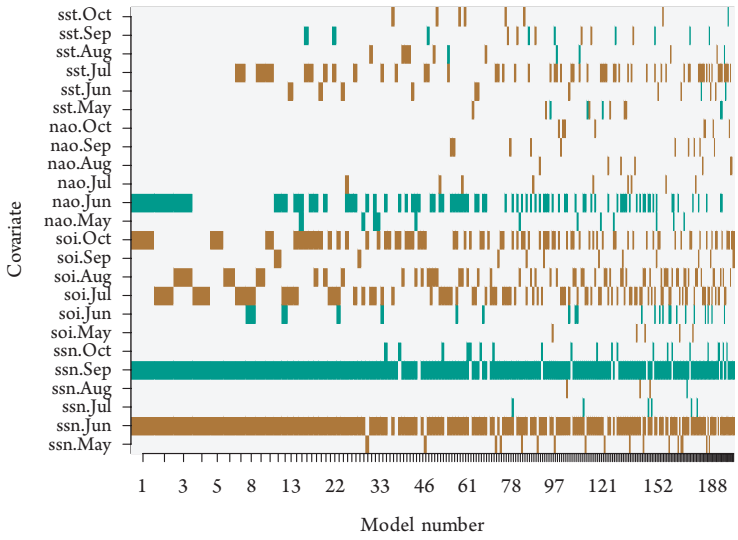

Figure 12.7 Covariates by model number.

where L is the likelihood evaluated at the parameter estimates, k is the number of parameters, and n is the number of years. BIC includes a penalty term $(k \ln(n))$, which makes it useful for comparing models with different sizes. If the penalty term is removed, BIC can be reduced simply by increasing the number of covariates. The BIC as a selection criterion results in choosing models that are parsimonious and asymptotically consistent, meaning that the model with the lowest BIC converges to the true model as the number of years increases.

You use a plot method to display the model coefficients (by sign) ordered by decreasing posterior probabilities as shown in Figure 12.7. Models are listed along the horizontal axis by decreasing posterior probability. Covariates in a model are shown with colored bars. A brown bar indicates a positive relationship with hurricane rate and a green bar indicates a negative relationship. Bar width is proportional to the model's posterior probability.

```
> imageplot.bma(mdls)
```

The plot makes it easy to see the covariates picked by the most probable models. They are the ones with the most consistent coloring from left to right across the image. A covariate with only a few gaps indicates that it is included in most of the higher probable models. These include September and June SSN, June NAO, July SST, and any of the months of July through September for the SOI.

You might ask why July SST is selected as a model covariate more often than August and September? The answer lies in the fact that when the hurricanes arrive in August and September, they draw heat from the ocean surface so the correlation between hurricane activity and SST weakens. The thermodynamics of hurricane intensification works against the statistical correlation. Said another way, July SST

better relates to an active hurricane season not because a warm ocean in July *causes* tropical cyclones in August and September, but because hurricanes in August and September cool the ocean slightly.

The SOI covariates get chosen frequently by the most probable models but with a mixture across the months of July through October. The posterior probability is somewhat higher for the months of June and October and smallest for August and September. Thunderstorms over the eastern equatorial Pacific during El Niño produce increased shear and subsidence across the Atlantic especially over the western Caribbean where during the months of July and October a relatively larger percentage of the North Atlantic hurricane activity occurs. Moreover, the inhibiting influence of El Niño might be less effective during the core months of August and September when, on average, other conditions tend to be favorable.

The sign on the September SSN parameter is negative indicating that the probability of a U.S. hurricane decreases with increasing number of sunspots. This result accords with the hypothesis that increases in UV radiation from an active sun (greater number of sunspots) warms the upper troposphere resulting in greater thermodynamic stability and a lower probability of a hurricane over the western Caribbean and Gulf of Mexico (Elsner and Jagger, 2008; Elsner et al., 2010). The positive relationship between hurricane probability and June SSN is explained by the direct influence the sun has on ocean temperature. Alternative explanations are possible especially in light of the role the solar cycle likely plays in modulating the NAO (Kodera, 2002; Ogi et al., 2003).

You can find the probability that a covariate irrespective of month is chosen by calculating the total posterior probability over all models that include this covariate. First, use the `substring` function on the covariate names given in the `bic.glm` object under `namesx` to remove the last four characters in each name. Also create a character string containing only the unique names.

```
> cn = substring(mdls$namesx, 1, 3)
> cnu = unique(cn)
```

Next create a matrix of logical values using the `outer` function, which performs an outer product of matrices and arrays. Also assign column names to the matrix.

```
> mn = outer(cn, cnu, "==")
> colnames(mn) = cnu
```

Next perform a matrix multiplication of the matching names with the matrix of logical entries indicating which covariate was chosen. This returns a matrix with dimensions of number of models by number of covariate types. The matrix entries are the number of covariates in each model of that type. Finally multiply the posterior probabilities given under `postprob` by a logical version of this matrix using the condition that the covariate type is included.

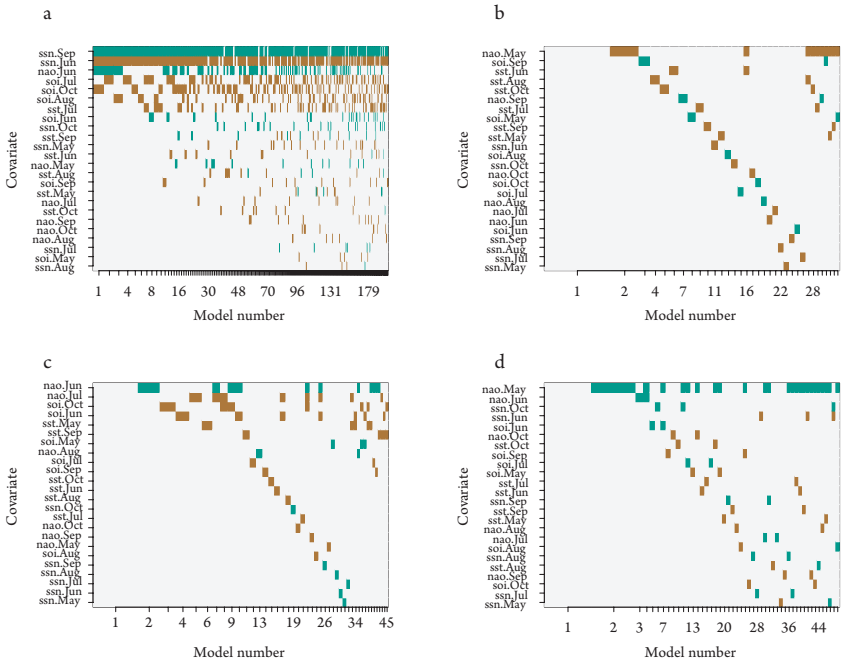

Figure 12.8 Covariates by model number. (a) Actual and (b–d) permuted series.

```
> xx = mdls$which %*% mn
> pc = 100 * mdls$postprob %*% (xx > 0)
> round(pc, 1)
      sst  nao  soi  ssn
[1,] 39.7 51.9 98.5 98.5
```

You see that the SOI and sunspot number have the largest probabilities at 98.5 percent while the NAO and SST have posterior probabilities of 51.9 percent and 39.7 percent, respectively. The lower probability of choosing the NAO reflects the rather large intraseasonal variability in this covariate as seen in Figure 12.6.

It is informative to compare the results of your BMA with a BMA performed on a random series of counts. Here you do this by resampling the actual hurricane counts. The randomization results in the same set of counts, but the counts are placed randomly across the years. The random series together with the covariates are used as before and the results are mapped in Figure 12.8.

The comparison shows that your set of covariates has a meaningful relationship with U.S. hurricane activity. There are fewer models chosen with the randomized data sets and the number of variables included in the set of most probable models is lower. In fact the averaged number of variables in the 20 most probable models is 4, which compares with an average of only one for the three randomized series. Moreover, there is little consistency in the variable selected from one model to the next as it should be with randomized data.

12.3.4 Consensus Hindcasts

As seen the BMA procedure assigns a posterior probability to a set of the most probable models. Each model can be used to make a prediction. But which forecast should you believe? Fortunately, no choice is necessary. Each model makes a prediction and then the forecasts are averaged. The average is weighted where the weights are proportional to the model's posterior probability. Here you assume perfect knowledge of the covariates and hindcasts are made in-sample. For an actual forecast situation, this is not available, but the method would be the same.

You use the prediction functions in **prediction.R** to hindcast annual rates, rate distributions, and posterior count distributions. First, source the code file, then compute the mean and standard deviation of the annual rate for each year. From the output, compute the in-sample average square error.

```
> source("prediction.R")
> ar = bic.poisson(mdls, newdata=mdls$x, simple=TRUE)
> sqrt(mean((ar[1, ] - mdls$y)^2))
[1] 1.36
```

Thus, on average, the consensus model results in a mean square error of 1.4 hurricanes per year.

Here you examine hindcast probabilities for the consecutive years of 2007 and 2008. You determine the posterior probabilities for the number of hurricanes for each year for hurricane numbers between zero and eight and display them using a side-by-side bar plot.

```
> yr1 = 2007; yr2 = 2008
> r1 = which(dat$Year==yr1)
> r2 = which(dat$Year==yr2)
> Pr = bic.poisson(mdls, newdata=mdls$x[c(r1, r2), ],
+   N=9)
> barplot(t(Pr), beside=TRUE, las=1,
+   xlab="Number of Hurricanes",
+   ylab="Probability", legend.text=
+   c(as.character(yr1), as.character(yr2)))
```

Results are shown in Figure 12.9. The vertical axis is the probability of observing h number of hurricanes. The model predicts a higher probability of at least one U.S. hurricane for 2008 compared with 2007. There is a 54 percent chance of three or more hurricanes for 2007 and a 57 percent chance of three or more hurricanes for 2008. There was one hurricane in 2007 and three hurricanes in 2008.

The consensus model hindcasts larger probabilities of an extreme year given the rate than would be expected from a Poisson process. That is, the consensus model is overdispersed with respect to a Poisson distribution. This is because model uncertainty is incorporated in the consensus hindcasts.

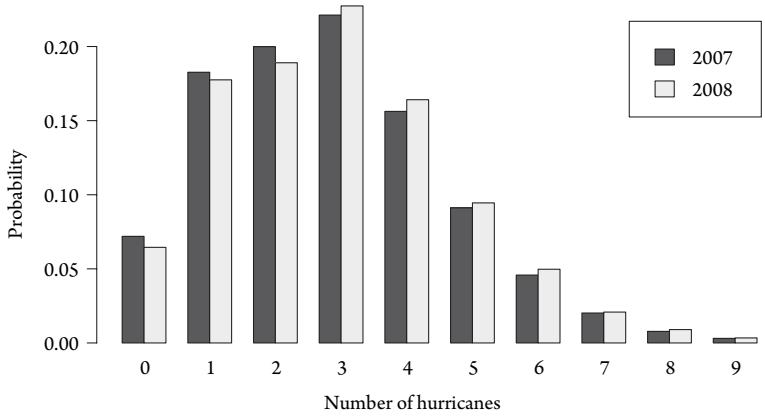

Figure 12.9 Forecasts from the consensus model.

A cross-validation of the BMA procedure is needed to get an estimate of how well the consensus model will do in predicting future counts. This is done in Jagger and Elsner (2010) using various scoring rules including the mean square error, the ranked probability score, the quadratic (Brier) score, and the logarithmic score. They find that the consensus model provides more accurate predictions than a procedure that selects a single best model using BIC or AIC irrespective of the scoring rule. The consensus forecast will not necessarily give you the smallest forecast error every year, but it will always provide a better assessment of forecast uncertainty compared to a forecast made from a single model. The BMA procedure provides a rational way for you to incorporate competing models into the forecast process.

12.4 SPACE–TIME MODEL

You save your most ambitious model for last. It draws on your knowledge of frequency models (Chapter 7), spatial models (Chapter 9), and Bayesian methods (Chapter 4). Substantial progress has been made in understanding and predicting hurricane activity on the seasonal and longer time scales over the basin as a whole. Much of this progress comes from statistical models described in this book.

However, significant gaps remain in our knowledge of what regulates hurricane activity *regionally*. Here your goal is a multilevel (hierarchical) statistical model to better understand and predict regional hurricane activity. Multilevel models are not new but have gained popularity with the growth of computing power and better software. However, they have yet to be employed to study hurricane climate. For a comprehensive modern treatment of statistics for spatiotemporal data, see Cressie and Wikle (2011). You begin with a set of local regressions.

12.4.1 Lattice Data

Here you use the spatial hexagon framework described in Chapter 9 to create space–time lattice data consisting of hurricane counts at each hexagon for each year and accompanying covariate information. Some of the covariate information is at the local

(hexagon) level and some of it is at the regional level (e.g., climate indices). Single-level models, including linear and generalized linear models, are commonly used to describe basin-wide hurricane activity, but multilevel models allow you to model variations in relationships at the individual hexagon level and for individual years. They are capable of describing and explaining within-basin variations.

Some data organization is needed. First input the hourly best-track data, the netCDF SST grids, and the annual aggregated counts and climate covariates that you arranged in Chapter 6. Specify also the range of years over which you want to model and make a copy of the best-track data frame.

```
> load("best.use.RData")
> load("ncdataframe.RData")
> load("annual.RData")
> years = 1886:2009
> Wind.df = best.use
```

Next define the hexagon tiling. Here you follow closely the work flow outlined in Chapter 9. Acquire the **sp** package. Then assign coordinates to the location columns and add geographic projection information as a coordinate reference system.

```
> require(sp)
> coordinates(Wind.df) = c("lon", "lat")
> coordinates(ncdataframe) = c("lon", "lat")
> ll = "+proj=longlat +ellps=WGS84"
> proj4string(Wind.df) = CRS(ll)
> proj4string(ncdataframe) = CRS(ll)
> slot(Wind.df, "coords")[1:3, ]
      lon lat
[1,] -94.8  28
[2,] -94.9  28
[3,] -95.0  28
```

With the `spTransform` function (**rgdal**), you change the geographic CRS to a Lambert conformal conic (LCC) planar projection using the parallels 15 and 45°N and a center longitude of 60°W.

```
> lcc = "+proj=lcc +lat_1=45 +lat_2=15 +lon_0=-60"
> require(rgdal)
> Wind.sdf = spTransform(Wind.df, CRS(lcc))
> SST.sdf = spTransform(ncdataframe, CRS(lcc))
> slot(Wind.sdf, "coords")[1:3, ]
         lon     lat
[1,] -3257623 3648556
[2,] -3266544 3651532
[3,] -3275488 3654496
```

The transformation does not rename the coordinates, but the values are the new LCC projected coordinates. You compare the bounding boxes to make sure that the cyclone data are contained in the SST data.

```
> bbox(Wind.sdf)
         min     max
lon -4713727 4988922
lat  1020170 8945682
> bbox(SST.sdf)
         min     max
lon -4813071 8063178
lat    78055 9185849
```

Next, generate the hexagons. First, sample the hexagon centers using the bounding box from the cyclone data. Specify the number of centers to be 250 and fix the offset so that the sampler will choose the same set of centers given the number of centers and the bounding box. Then create a spatial polygons object:

```
> hpt = spsample(Wind.sdf, type="hexagonal",
+   n=250, bb=bbox(Wind.sdf) * 1.2, offset=c(1, -1))
> hpg = HexPoints2SpatialPolygons(hpt)
```

This results in 225 hexagons each with an area of approximately 511,457 km^2.

Next, overlay the hexagons on the cyclone and SST locations separately.

```
> Wind.hexid = over(x=Wind.sdf, y=hpg)
> SST.hexid = over(x=SST.sdf, y=hpg)
```

This creates a vector containing the hexagon identification number for each hourly observation. The length of the vector is the number of observations. Similarly, for the SST data, the integer vector has elements indicating in which hexagon the SST value occurs.

Then use the `split` function to divide the data frame into groups defined by the hexagon number. The groups are saved as lists. Each list is a data frame containing information specific to the cyclones occurring in each hexagon. You do this for the cyclone and SST data.

```
> Wind.split = split(Wind.sdf@data, Wind.hexid)
> SST.split = split(SST.sdf@data, SST.hexid)
```

You find that the first hexagon contains five cyclone observations. To view a selected set of the columns from this data frame, type

```
> Wind.split[[1]][c(1:2, 5:7, 9, 11)]
          Sid Sn   Yr Mo Da Wmax DWmaxDt
3398      185  5 1878  9  1 50.0   0.467
3398.1    185  5 1878  9  1 49.4   0.484
33539.5 1168  6 1990  8 13 34.8   1.222
```

```
33540   1168  6 1990  8 13 35.0   1.005
42583   1442 19 2010 10 29 30.0   0.690
```

A given hexagon tends to capture more than one cyclone hour. To view a selected set of SST grid values from the corresponding hexagon, type

```
> SST.split[[1]][, 1:5]
   Y1854M01 Y1854M02 Y1854M03 Y1854M04 Y1854M05
1      24.7     26.2     26.6     26.3     25.5
2      24.7     26.3     26.7     26.2     25.3
3      24.7     26.4     26.7     26.1     25.0
58     25.6     26.8     26.9     26.7     26.2
```

The hexagon contains four SST grid values and there are 1,871 months as separate columns starting in January 1854 (`Y1854M01`). Next, reassign names to match those corresponding to the hexagon identifications.

```
> names(Wind.split) = sapply(hpg@polygons, function(x)
+   x@ID)[as.numeric(names(Wind.split))]
> names(SST.split) = sapply(hpg@polygons, function(x)
+   x@ID)[as.numeric(names(SST.split))]
```

There are hexagon grids with cyclone data over land areas (no SST data), and there are areas over the ocean where no cyclones occurr. Thus, you subset each to match hexagons having cyclones and SST data.

```
> Wind.subset = Wind.split[names(Wind.split) %in%
+   names(SST.split)]
> SST.subset = SST.split[names(SST.split) %in%
+   names(Wind.split)]
```

The function `%in%` returns a logical vector indicating whether there is a match for the names in `Wind.split` from the set of names in `SST.split`. The data sets are now in synch. There are 109 hexagons with both cyclone and SST data. Note that for the cyclone data, you could subset `best.use` on `M==FALSE` to remove cyclone observations over land.

Next, compute the average SST within each hexagon by month and save them as a data frame.

```
> SST.mean = data.frame(t(sapply(SST.subset,
+   function(x) colMeans(x))))
> head(SST.mean)[1:5]
     Y1854M01 Y1854M02 Y1854M03 Y1854M04 Y1854M05
ID8      26.9     27.0     27.0     27.3     27.6
ID9      27.0     27.2     27.2     27.4     27.4
ID18     27.9     27.9     28.0     28.2     28.3
ID19     26.9     26.5     26.5     27.1     28.2
```

```
ID20     26.8     26.3     26.1     26.9     27.9
ID21     27.0     26.5     26.3     27.0     27.9
```

The data frame is organized with the hexagon identifier as the row (observation) and consecutive months as the columns (variables). Thus, there are 109 rows and 1,871 columns. Your interest in SST is more narrowly focused on the months of August through October when hurricanes occur. To generate these values by year, type

```
> SSTYearMonth = strsplit(substring(colnames(SST.mean),
+   2), "M")
> SSTYear = as.numeric(sapply(SSTYearMonth,
+   function(x) x[1]))
> SSTMonth = as.numeric(sapply(SSTYearMonth,
+   function(x) x[2]))
> SSTKeep = which(SSTMonth %in% c(8, 9, 10))
> SSTYear = SSTYear[SSTKeep]
```

The vector `SSTKeep` lists the column numbers corresponding to August, September, and October for each year and the vector `SSTYear` is the set of years for those months.

Next, subset `SST.mean` by `SSTKeep` and then compute the August–October average for each year.

```
> SST.mean.keep = SST.mean[, SSTKeep]
> SST.mean.year = sapply(unique(SSTYear), function(x)
+   as.vector(rowMeans(SST.mean.keep[,
+   which(x==SSTYear)])))
> dimnames(SST.mean.year) =
+   list(id=rownames(SST.mean.keep),
+   Year=paste("Y",unique(SSTYear), sep=""))
> SST.mean.year.subset = SST.mean.year[,
+   paste("Y", years, sep="")]
> SST.pdf = SpatialPolygonsDataFrame(
+   hpg[rownames(SST.mean.year.subset)],
+   data.frame(SST.mean.year.subset))
```

The data slot in the spatial polygon data frame `SST.pdf` contains the average SST during the hurricane season for each year. To list the first few rows and columns, type

```
> slot(SST.pdf, "data")[1:5, 1:6]
     Y1886 Y1887 Y1888 Y1889 Y1890 Y1891
ID8   28.4  28.2  28.5  28.4  27.7  28.4
ID9   28.0  27.8  28.1  28.0  27.3  28.1
ID18  27.7  28.0  28.5  27.5  27.6  27.7
ID19  28.5  28.4  28.6  28.5  27.9  28.5
ID20  28.4  28.4  28.1  28.2  27.6  28.6
```

You plot the SST data as you did in Chapter 9 using the `spplot` method. First, obtain the map borders in geographic coordinates and project them using the same CRS as your data.

```
> require(maps)
> require(maptools)
> require(colorRamps)
> cl = map("world", xlim=c(-120, 20),
+   ylim=c(-10, 70), plot=FALSE)
> clp = map2SpatialLines(cl, proj4string=CRS(ll))
> clp = spTransform(clp, CRS(lcc))
> l2 = list("sp.lines", clp, col="darkgray")
```

Then, obtain the color ramps and plot the values from the year 1959, for example.

```
> spplot(SST.pdf, "Y1959", col="white",
+    col.regions=blue2red(20), pretty=TRUE,
+    colorkey=list(space="bottom"),
+    sp.layout=list(l2),
+    sub="Sea Surface Temperature (C)")
```

Your plot shows how the data are organized.

Next, you need to generate a data set of cyclones that correspond to these hexagons in space and time. First, generate a list of the cyclones maximum intensity by hexagon using your `get.max` function and count the number of cyclones per hexagon per year.

```
> source("getmax.R")
> Wind.max = lapply(Wind.subset, function(x)
+   get.max(x, maxfield="WmaxS"))
> Wind.count = t(sapply(Wind.max, function(x)
+   table(factor(subset(x, WmaxS >= 33 &
+   Yr %in% years)$Yr, level=years))))
> colnames(Wind.count) = paste("Y",
+   colnames(Wind.count), sep="")
> Wind.count = data.frame(Wind.count)
> Wind.pdf = SpatialPolygonsDataFrame(
+   hpg[rownames(Wind.count)], Wind.count)
```

As an example, to list the hurricane counts by hexagon for 1959, type

```
> slot(Wind.pdf, "data")["Y1959"]
```

12.4.2 Local Independent Regressions

With your annual hurricane counts and seasonal SST collocated spatially, you build local (hexagon-level) independent Poisson regressions (LIPR). Here "independent"

refers to separate regressions one for each hexagon. Along with the hexagon-level SST variable, you include the SOI as a covariate. The SOI varies by year but not by hexagon. The SOI covariate was organized in Chapter 6 and saved in *annual.RData*.

Load the annual climate covariates and subset the SOI and SST columns for the years specified in the previous section. These are your regional covariates.

```
> load("annual.RData")
> Cov = subset(annual, Year %in% years)[,
+   c("soi", "sst", "Year")]
```

Then create a data frame of your hexagon-level SST covariate and the hurricane counts from the data slots in the corresponding spatial polygon data frames.

```
> LSST = slot(SST.pdf, "data")
> Count = slot(Wind.pdf, "data")
```

Here you build 109 separate regressions, one for each hexagon. Your annual count, indicating the number of hurricanes whose centers passed through, varies by hexagon and by year and you have local SST and regional SOI as covariates. Both are averages over the months of August–October.

The model is a Poisson regression with a logarithmic link function.

```
> lipr = lapply(1:nrow(Count), function(i)
+   glm(unlist(Count[i, ]) ~ unlist(LSST[i, ]) +
+   Cov$soi + Cov$Year, family="poisson"))
```

The standardized coefficients indicate the strength of the relationship between the covariate and the annual count. The coefficient are saved for each hexagon in the matrix `zvals` that you turn into a spatial polygon data frame.

```
> zvals = t(sapply(lipr, function(x)
+   summary(x)$coef[, 3]))
> rownames(zvals) = rownames(Count)
> colnames(zvals) = c("Intercept", "Local.SST",
+   "SOI", "Year")
> zvals.pdf = SpatialPolygonsDataFrame(
+   hpg[rownames(zvals)], data.frame(zvals))
```

To map the results, first generate a color ramp function, then use the `spplot` method.

```
> al = colorRampPalette(c("blue","white","red"),
+   space="Lab")
> spplot(zvals.pdf, c("Local.SST","SOI"),
+    col.regions=al(20), col="white",
+    names.attr=c("SST", "SOI"),
+    at=seq(-5, 5),
+    colorkey=list(space="bottom",
```

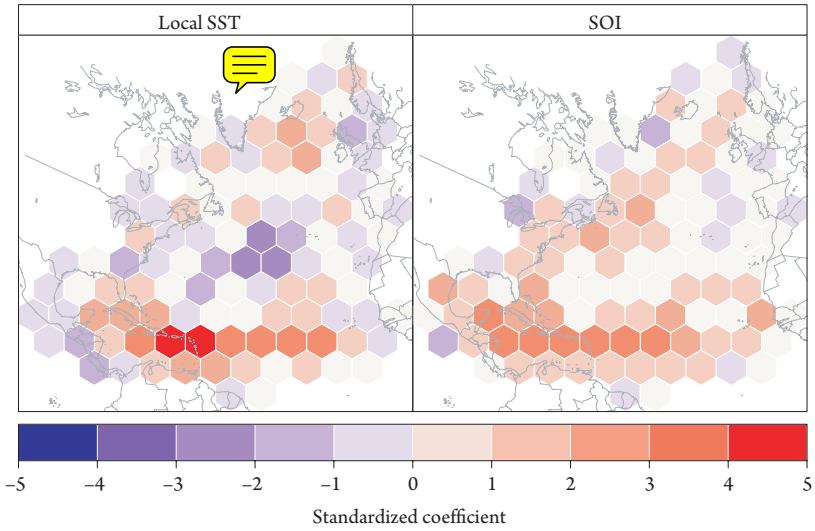

Figure 12.10 Poisson regression coefficients of counts on local SST and SOI.

```
+     labels=paste(seq(-5, 5))),
+     sp.layout=list(l2),
+     sub="Standardized Coefficient")
```

The maps are shown in Figure 12.10. The standardized coefficient can be interpreted as a hypothesis test, one for each of the hexagons. Values more than 2 in absolute value greatly indicate a significant relationship between the covariate and hurricane probability. The probability of a hurricane is higher where the ocean is warm and when SOI is positive (La Niña conditions), a result you would anticipate from your basin-wide models (Chapter 7). The SST and SOI effects are strongest over the central and western North Atlantic at low latitudes. Curiously, the SST effect is muted along much of the eastern coast of the United States and along portions of the Mexican coast northward through Texas. This might explain why, despite warming seas, the U.S. hurricane counts do not show an increase over the past century and a half.

Your model contains the year as a covariate to address changes in occurrence rates over time. The exponent of the coefficient on the year term is the factor by which the occurrence rates are changing per year. The factor is shown for each grid in Figure 12.11. The factors range between 0.98 and 1.02 annually depending on location. One indicates no change, less than one a decreasing trend, and greater than one an increasing trend. Since the data cover the period beginning in 1886, the increasing trends over the central and eastern North Atlantic might be because of improving surveillance. However, the downward trend over a large part of the Caribbean Sea into the Gulf of Mexico is intriguing. It might be related to increasing wind shear or continental aerosols. In fact, you can see the downward trend along parts of the U.S. coastline from Louisiana to South Carolina and over New England.

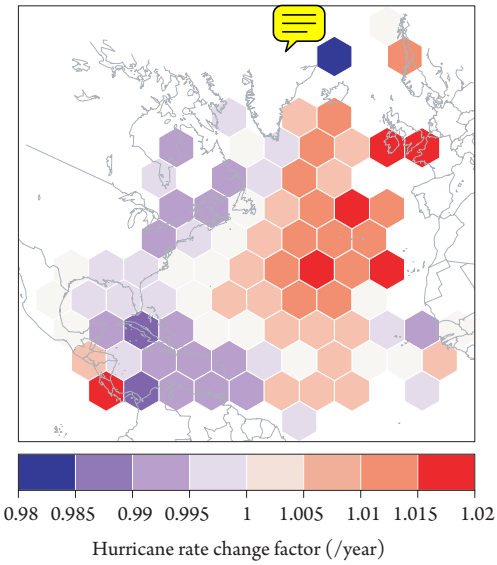

Figure 12.11 Factor by which hurricane rates have changed per year.

Model residuals, defined as the observed count minus the predicted rate for a given year and hexagon, can also be mapped. Here you map the residuals for the 2005 hurricane season. First, compute the residuals by typing

```
> preds = t(sapply(lipr, function(x)
+   predict(x, type="response")))
> rownames(preds) = rownames(Count)
> err2005 = Count[, "Y2005"] - preds[, "Y2005"]
> err.pdf = SpatialPolygonsDataFrame(
+   hpg[names(err2005)], data.frame(err2005))
```

Then select a color ramp function and create a choropleth map with the `spplot` method.

```
> al = colorRampPalette(c("blue", "white", "red"),
+   space="Lab")
> spplot(err.pdf, c("err2005"), col="white",
+   col.regions=al(20), at=seq(-5, 5, 1),
+   colorkey=list(space="bottom",
+   labels=paste(seq(-5, 5, 1))),
+   sp.layout=list(l2),
+   par.settings=list(fontsize=list(text=10)),
+   sub="Observed [count] - Predicted [rate]")
```

Figure 12.12 shows where the model over-(blues) and under-(reds) predicts for the 2005 hurricane season. The model underpredicted the large amount of hurricane activity over the western Caribbean and Gulf of Mexico.

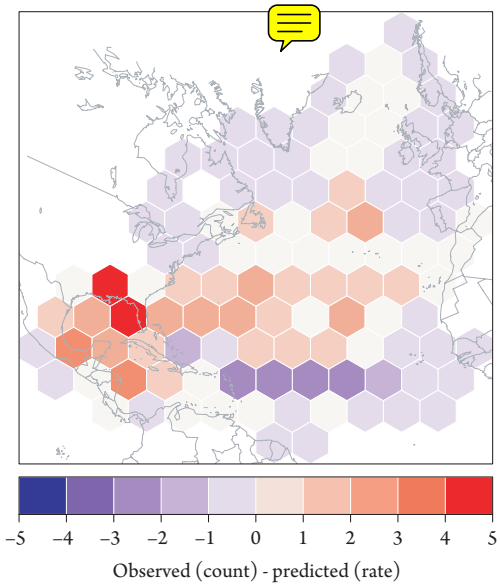

Figure 12.12 Model residuals for the 2005 hurricane season.

12.4.3 Spatial Autocorrelation

Note the residuals tend to be spatially correlated. Residuals in neighboring hexagons tend to be more similar than residuals in hexagons farther away. To quantify the degree of spatial correlation, you create a weights matrix indicating the spatial neighbors for each hexagon (see Chapter 9). The **spdep** package (Bivand et al., 2011a) has functions for creating weights based on contiguity neighbors. First, you use the `poly2nb` function (**spdep**) on the spatial polygons data frame to create a contiguity-based neighborhood list object.

```
> require(spdep, quietly=TRUE)
> hexnb = poly2nb(err.pdf)
```

The list is ordered by hexagon number. The first hexagon has three neighbors; hexagon numbers 2, 7, and 8. Hexagon numbers increase to the west and north. A hexagon has at most six contiguous neighbors. Hexagons at the borders have fewer neighbors. A graph of the hexagon connectivity, here defined by the first-order contiguity, is made by typing

```
> plot(hexnb, coordinates(err.pdf))
> plot(err.pdf, add=TRUE)
```

A summary method applied to the neighborhood list (`summary(hexnb)`) reveals the average number of neighbors and the distribution of connectivity among the hexagons.

You turn the neighborhood list object into a `listw` object using the `nb2listw` function that adds weights to the neighborhood list. The `style` argument determines the weighting scheme. With the argument value set to `W`, the weights are the inverse of the number of neighbors.

```
> wts = nb2listw(hexnb, style="W")
```

Next, you quantify the amount of spatial correlation through the value of Moran's *I*. This is done using the function `moran` (**spdep**). The first argument is the variable of interest followed by the name of the `listw` object. Also you need the number of hexagons and the global sum of the weights, which is obtained using the `Szero` function.

```
> n = length(err.pdf$err2005)
> s = Szero(wts)
> mI = moran(err.pdf$err2005, wts, n=n, S0=s)$I
```

The function returns a value for Moran's *I* of 0.39, which indicates spatial autocorrelation in model residuals for 2005. The expected value of Moran's *I* under the hypothesis of no spatial autocorrelation is $-1/(n-1)$, where n is the number of hexagons. This indicates that the model can be improved by including a term for the spatial correlation. Other years have more or less residual autocorrelation, which might be interesting to examine in more detail.

12.4.4 BUGS Data

Next you build a spatial regression model. The model allows you to borrow information from neighboring hexagons. The model is motivated by the fact that the residuals from your nonspatial regression above are spatially correlated and hurricanes are infrequent in most hexagons. The model is written in BUGS (see Chapter 4). Here you run BUGS (WinBUGS or OpenBUGS) outside of R.

As preparation you first convert your neighborhood list object to BUGS format. Then you gather the hurricane counts and covariates as lists and put them together in the object you call `BUGSdata`.

```
> hexadj = nb2WB(hexnb)
> BUGSData = c(list(S=nrow(Wind.count),
+    T=ncol(Wind.count), h=as.matrix(Wind.count),
+    SST=as.matrix(SST.mean.year.subset),
+    SOI=Cov$soi), hexadj)
```

For each hexagon (`S` of them) and year (`T` of them), there is a count (`N`) that indicates the number of hurricanes. The associated covariates are the SOI and local SST. The local SST is constructed by averaging the August–October monthly gridded SST over each hexagon for each year. For each hexagon, the neighborhood lists indicate the neighboring hexagons by ID (adjacency `adj`), the weights for the neighbors (weights `wts`), and the number of neighbors (number `num`).

Finally, you use `writeDatafileR` from the file **writedatafileR.R** to write an ASCII text representation of this to your working directory.

```
> source("writedatafileR.R")
> writeDatafileR(BUGSData,"BugsData.txt")
```

12.4.5 MCMC Output

Counts in each hexagon for each year are described by a Poisson distribution (`dpois`) with a rate that depends on hexagon and year (`lambda`). The logarithm of the rate is conditional on local SST and SOI. The error term (`error`) and the local effect terms include structured and unstructured components following Besag et al. (1991), where the structured component is an intrinsic conditional autoregressive (ICAR) specification. The adjacency matrix (`adj`), which gives your contiguity neighborhood as defined earlier is part of the ICAR specification.

The BUGS code for the model is

```
___BUGS code___
model {
for(hx in 1:S) {
  for(yr in 1:T) {
# Poisson likelihood for observed counts
   h[hx, yr] ~ dpois(lambda[hx, yr])
   log(lambda[hx, yr]) <- sst[hx] * SST[hx, yr] +
                          soi[hx] * SOI[yr] +
                          error[hx]
      }
# Error terms
   error[hx] <- u.error[hx] + s.error[hx]
   sst[hx] <- u.sst[hx] + s.sst[hx]
   soi[hx] <- u.soi[hx] + s.soi[hx]
# Unstructured errors
   u.error[hx] ~ dnorm(int, tau)
   u.sst[hx] ~ dnorm(int.sst, tau.sst)
   u.soi[hx] ~ dnorm(int.soi, tau.soi)
   }
# Structured errors
s.error[1:S] ~ car.normal(adj[], weights[],
                          num[], tau.s)
s.sst[1:S] ~ car.normal(adj[], weights[],
                          num[], tau.s.sst)
s.soi[1:S] ~ car.normal(adj[], weights[],
                          num[], tau.s.soi)
```

```
# Priors
int ~ dflat()
int.sst ~ dflat()
int.soi ~ dflat()
tau ~ dgamma(.5, .005)
tau.sst ~ dgamma(.5, .005)
tau.soi ~ dgamma(.5, .005)
tau.s ~ dgamma(.5, .005)
tau.s.sst ~ dgamma(.5, .005)
tau.s.soi ~ dgamma(.5, .005)
}
```

Think of the model as a directed acyclic graph (DAG) as shown in Chapter 4. Nodes are the parameters and data and the arrows indicate conditional dependency, either stochastic (~) or deterministic (<-).

You use uninformative (flat) priors for the intercept terms and a gamma distribution for priors on the precisions. A shape parameter of 0.5 and a scale of 0.005 translates to a 1 percent probability that the standard deviation is less than 0.04 and a 1 percent probability that the standard deviation is greater than .8, respectively. It would also be reasonable to combine the models into a multivariate CAR model, with a Wishart prior, having 4 degrees of freedom and a diagonal of 0.05.

Open BUGS (outside R) and copy the code to a new BUGS document saving it as (`BUGSmodel.odc`). Open the `BUGSdata.txt` file and copy the entire file to your BUGS document. Finally, copy the initial values below into the same document.

```
___Initial Values___
list(int=0, int.sst=0, int.soi=0, tau=100, tau.sst=100,
  tau.soi=100, tau.s=100, tau.s.sst=100, tau.s.soi=100)
list(int=.5, int.sst=.5, int.soi=.5, tau=100, tau.sst=100,
  tau.soi=100, tau.s=100, tau.s.sst=100, tau.s.soi=100)
```

It might be necessary to specify high values for the precisions since you need to have BUGS generate realistic initial samples for the uninitialized parameters. You use two sets of initial values (two separate `list` objects) to help monitor and diagnose convergence toward the posterior density. Each set will result in a separate chain of samples. Here you specify separate chains using a different value for the intercept terms.

In BUGS you first select Model > Specification to open the specification tool. Highlight the word `model` in your document then select `check` model. It will tell you that your model is syntactically correct in the lower left corner of the window. Next, highlight the word `list` in front of your data list and select `load data`. Then change the number of chains to two and select `compile`. It will tell you that the data are loaded and the model is compiled. Next, highlight the word `list` in front of your

first set of initial values and select `load inits`. It will tell you that this chain contains uninitialized values. Repeat for your second set of initial values. It will again tell you about uninitialized values. Finally, select `gen inits` to initialize the remaining values (for both chains).

Next you tell BUGS what model parameters (nodes) you want to monitor. Select Inference > Samples to open the sample monitor tool. In the `node` window, type `sst` and then select `set`. These parameters are the coefficients on the SST variable. Repeat for the coefficients on the SOI variable by typing `soi` and selecting `set`. Note that there are 109 SST and SOI coefficients, one of each for each hexagon.

Next, you select Model > Update to open the update tool. In the `update` window, type 5000, and in the `refresh` window, type 10. Although you are monitoring only the SST and SOI coefficients, each update (iteration) represents new values for all parameters in your model using an MCMC algorithm (see Chapter 4). Select `update`. This will take time. The refresh number indicates the progress in intervals of 10 updates.

After the MCMC has finished updating, you output the monitored parameter values from both chains. In the `node` window, scroll to `sst`, then select `coda`. CODA is a suite of functions for analyzing outputs from BUGS software. Save the samples and index files separately (use the extension `txt`). The index file provides the order of the samples.

12.4.6 Convergence and Mixing

Before using your samples for inference, you need to check a few things. Starting from your initial values, the MCMC algorithm produces a new set of values (the first update sample) for each parameter (node). In most cases (except in simple models), these new values will not represent the posterior density. As updating continues the MCMC samples are guaranteed to *converge* to a stationary distribution representing samples from your posterior. It is useful to know the minimum number of updates needed until convergence. The period before convergence is called "burn-in", where the samples are moving away from the initial set of values and toward the posterior distribution.

You can analyze your MCMC samples in BUGS directly, but there are more options in R. Input the BUGS-outputted CODA files corresponding to the SST parameter using the `read.coda` function (**coda**). The first argument is the name of the samples file and the second is the name of the index file. Your new objects (`chain1` and `chain2`) have class `mcmc`.

```
> require(coda)
> chain1 = read.coda("Chain1.txt",
+   "Index.txt", quiet=TRUE)
> chain2 = read.coda("Chain2.txt",
+   "Index.txt", quiet=TRUE)
```

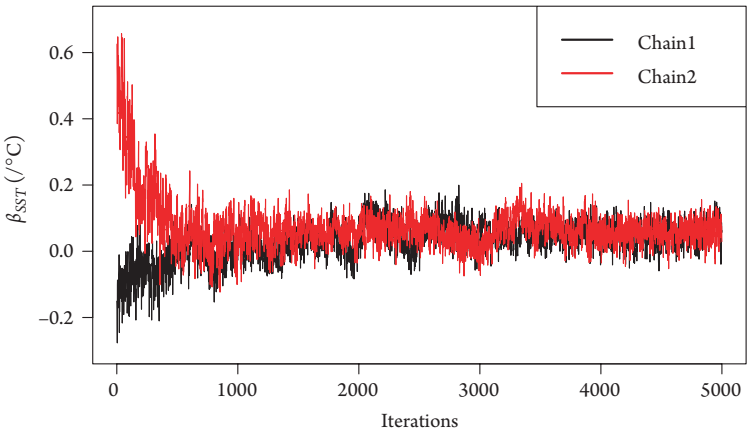

Figure 12.13 MCMC from a space–time model of cyclone counts in hexagon one.

To plot the MCMC samples of the SST coefficient in the first hexagon (hexagons are ordered from southwest to northeast) from both chains, type

```
> traceplot(chain1[, "sst[1]"], ylim=c(-.3, .7))
> traceplot(chain2[, "sst[1]"], col="red", add=TRUE)
```

This generates a graph showing the sequence of values (trace plot) from the first (black) and second (red) chains (Fig. 12.13). Values fluctuate from one iteration to the next but tend toward a stable distribution after about 3,000 updates. This tendency toward convergence is even more apparent by comparing the two trace plots. Initially, the values from chain one are quite different from those of chain two, but after about 4,000 iterations the distributions are visually indistinguishable. From this analysis, you estimate that convergence requires about 4,000 iterations.

You quantify MCMC convergence with the potential scale reduction factor (PSRF) proposed by Gelman and Rubin (1992). After convergence, your two chains starting with different initial conditions should represent samples from the same distribution. You assess this by comparing the mean and variance of each chain to the mean and variance of the *combined* chain. Specifically, with two chains, the between-chain variance B/n and pooled within-chain variance W are defined by

$$\frac{B}{n} = \frac{1}{2-1}\sum_{j=1}^{2}(\bar{s}_{j.} - \bar{s}_{..})^2 \tag{12.10}$$

and

$$W = \frac{1}{2(n-1)}\sum_{j=1}^{2}\sum_{t=1}^{n}(s_{jt} - \bar{s}_{j.})^2 \tag{12.11}$$

where s_{jt} is the parameter value of the tth sample in the jth chain, $\bar{s}_{j.}$ is the mean of the samples in j, and $\bar{s}_{..}$ is the mean of the combined chains.

By taking the sampling variability of the combined mean into account, you get a pooled estimate for the variance:

$$\hat{V} = \frac{n-1}{n}W + \frac{B}{n} \tag{12.12}$$

Then an estimate $\hat{R}$ for PSRF is obtained by dividing the pooled variance with the pooled within-chain variance,

$$\hat{R} = \frac{\hat{V}}{W} = \frac{n-1}{n} + \frac{B}{nW} \tag{12.13}$$

If the chains have not converged, Bayesian credible intervals based on the *t*-distribution are too wide and have the potential to shrink by the PSRF. PSRF is sometimes called the shrink factor.

A value for $\hat{R}$ is available for each monitored node using the `gelman.diag` function (**coda**). Values substantially above 1 indicate lack of convergence. For example, to get the PSRF estimate on the SST parameter for the first hexagon, type

```
> gelman.diag(mcmc.list(chain1[, "sst[1]"],
+   chain2[, "sst[1]"]))
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]       1.01       1.04
```

By default, only the second half of the chain is used. The point estimate indicates near convergence consistent with the evidence in the trace plots. To see the evolution of $\hat{R}$ as the number of iterations increase, type

```
> gelman.plot(mcmc.list(chain1[, "sst[1]"],
+   chain2[, "sst[1]"]))
```

The plot shows convergence after about 3,000 iterations.

Convergence does not guarantee that your samples have visited all (or even a large portion) of the posterior density. The speed with which your samples work their way through the posterior (mixing) depends on how far they advance from one update to the next. Mixing efficiency is inversely related to the between-sample correlation decay as a function of sample lag. Sharp decay of the correlation indicates mixing is efficient.

The autocorrelation function shows the correlation as a function of consecutive sample lag. Here you use the `acf` function on the chain to examine the correlation by typing

```
> acf(chain1[, "sst[1]"], ci=0, lag.max=2000)
```

The results for the SST coefficient in hexagon 1 and hexagon 60 are shown in Figure 12.14. Depending on the hexagon, the decay of the positive correlation drops below 0.1 after several hundred samples.

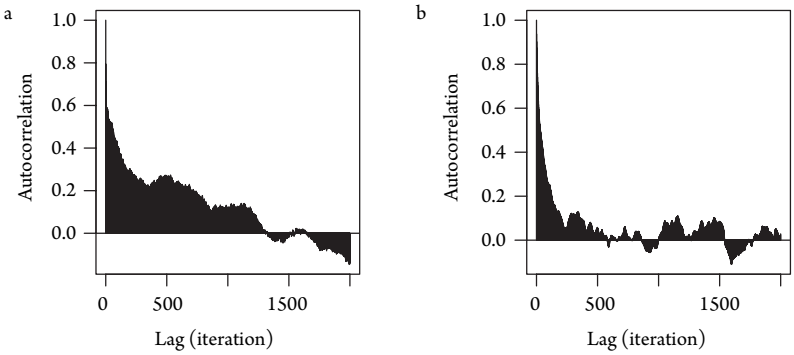

Figure 12.14 Autocorrelation of the SST coefficient in hexagon (a) 1 and (b) 60.

You use the effective chain size to quantify the decay. Because of the between-sample correlation, your effective chain size is less than the 5K updates. You use the `effectiveSize` function (**coda**) to estimate the effective chain size by typing

```
> effectiveSize(chain1[, "sst[1]"])
var1
  25
> effectiveSize(chain1[, "sst[60]"])
var1
59.7
```

The effective chain size is an estimate of how many independent samples you have given the amount of autocorrelation. The slower the decay of the autocorrelation function, the lower the effective size. For chain one, the effective size on the SST coefficient is 25 for the first hexagon and 60 for hexagon 60.

These numbers are too small to make reliable inferences, so you need more updates. Your goal is 1,000 independent samples. Since a conservative estimate of your effective sample size is 25 in 5K updates, you need 200K updates to reach your goal.

12.4.7 Updates

First return to BUGS. Next create a single chain and generate 205K updates. This time you monitor both the SST and SOI coefficients. After updating (this will take several hours),[4] you output every 200th sample over the last 200K updates. This is done with the sample monitor tool by setting `beg` equal to 5001 and `thin` equal to 200. Save the index files and chain values from both coefficients and read them into R.

[4] 200K updates took 10.5 hr on a 2 × 2.66 GHz dual-core Intel Xeon processor running on OS X version 10.6.8

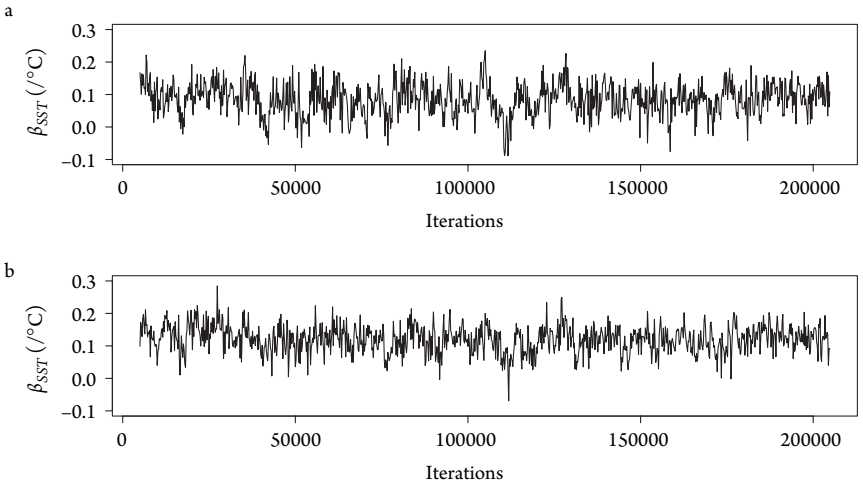

Figure 12.15 MCMC trace of the SST coefficients for hexagon (a) one and (b) 60.

```
> sstChain = read.coda("SSTChain.txt",
+   "SSTIndex.txt", quiet=TRUE)
> soiChain = read.coda("SOIChain.txt",
+   "SOIIndex.txt", quiet=TRUE)
```

You create a trace plot of your samples for hexagon one by typing

```
> traceplot(sstChain[, "sst[1]"], ylim=c(-.1, .3))
```

Note the apparent stationarity (see Fig. 12.15). Also note that since you saved only every 200th sample, the variation from one value to the next is much higher.

Now you use these samples to make inferences. For example, the probability that the SST coefficient for hexagon 1 is greater than zero is obtained by typing

```
> sum(sstChain[, "sst[1]"]>0)/
+   length(sstChain[, "sst[1]"]) * 100
[1] 95.2
```

You interpret the coefficient of the Poisson regression as a factor increase (or decrease) in the rate of occurrence per unit of the explanatory variable, given that the other covariates are held constant. This is a relative risk or the ratio of two probabilities. The relative risk of hurricanes per degree celsius in hexagon 1 is estimated using the posterior mean as

```
> exp(mean(sstChain[, "sst[1]"]))
[1] 1.09
```

A relative risk of one indicates no increase or decrease in occurrence probability relative to the long-term average. So you interpret the value of 1.09 to mean a 9 percent

increase per degree celsius and only a 4.8 percent chance that the relative risk is one (no change).

12.4.8 Relative Risk Maps

Here you map the relative risk of hurricanes. This time you estimate it using the posterior median and you do this for all hexagons by typing

```
> RRsst = exp(apply(sstChain, 2, function(x)
+   median(x)))
```

Next, create a spatial data frame of the relative risk with row names equal to your hexagon identification numbers.

```
> RRsst.df = data.frame(RRsst)
> rownames(RRsst.df) = rownames(Wind.count)
> RRsst.pdf = SpatialPolygonsDataFrame(
+   hpg[rownames(RRsst.df)], RRsst.df)
```

Then choose a color ramp and create a choropleth map.

```
> al = colorRampPalette(c("#FEE8C8", "#FDBB84",
+   "#E34A33"), space="Lab")
> spplot(RRsst.pdf, col="white", col.regions=al(20),
+   at=seq(1, 1.25, .05), colorkey=list(space="bottom",
+   labels=paste(seq(1, 1.25, .05))),
+   sp.layout=list(l2),
+   sub="Relative Hurricane Risk [/C]")
```

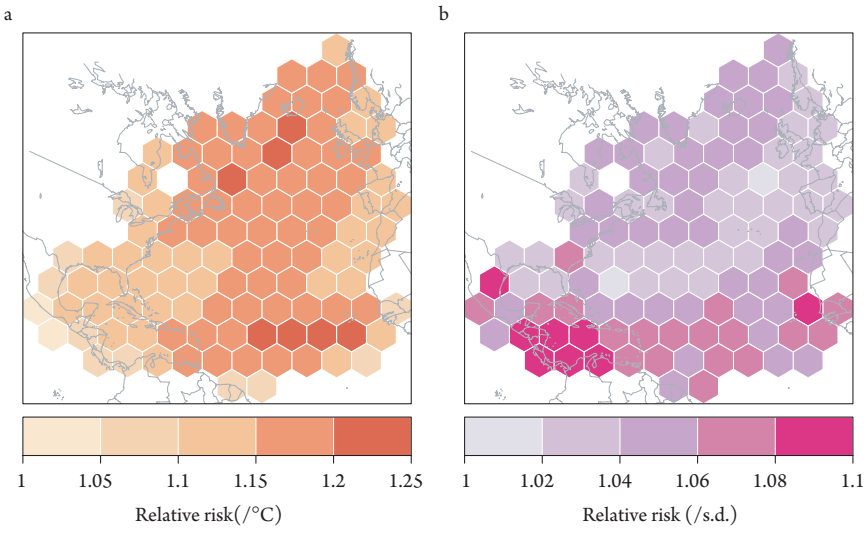

Figure 12.16 Hurricane risk per change in (a) local SST and (b) SOI.

Figure 12.16 maps the hurricane risk relative to a per degree celsius change in local SST and a per standard deviation change in SOI. Hurricane occurrence is most sensitive to rising ocean temperatures over the eastern tropical North Atlantic and less so across the western Caribbean Sea, Gulf of Mexico, and across much of the coast lines of Central America, Mexico, and the United States. In comparison, hurricane occurrence is most sensitive to ENSO over much of the Caribbean and less so over the central and northeastern North Atlantic.

This chapter demonstrated Bayesian models for hurricane climate research. We began by showing how to combine information about hurricane counts from the modern and historical cyclone archives to get a baseline estimate of future activity over the next several decades. We then showed how to create a Bayesian model for seasonal forecasts, where the model is based on an MCMC algorithm that allows you to exploit the older, less reliable cyclone information. We showed how to create a consensus model for seasonal prediction based on Bayesian model averaging. The approach circumvents the need to choose a single "best" model. Finally, we showed how to create a hierarchical model for exploiting the space–time nature of hurricane activity. Widespread application of Bayesian hierarchical models will undoubtedly lead to a better understanding of hurricane climate. In the final chapter, we consider a few practical applications of our statistical approach to hurricane climatology.