11

CLUSTER MODELS

"There are in fact two things, science and opinion; the former begets knowledge, the latter, ignorance."
—Hippocrates

A cluster is a group of the same or similar events close together. Clusters arise in hurricane origin locations, tracks, and landfalls. In this chapter, we look at how to analyze and model clusters. We divide the chapter into time, space, and feature clustering. Among climatologists feature clustering is perhaps the most well-known. We begin by showing you how to detect and model time clusters.

11.1 TIME CLUSTERS

Hurricanes form over certain regions of the ocean. Consecutive hurricanes from the same area often take similar paths. This grouping, or clustering, increases the potential for multiple landfalls above what you expect from random events.

A statistical model for landfall probability captures clustering through covariates like the North Atlantic Oscillation (NAO), which relates a steering mechanism (position and strength of the subtropical high pressure) to coastal hurricane activity. But there could be additional time correlation not related to the covariates. A model that does not account for this extra variation will underestimate the potential for multiple hits in a season.

Following Jagger and Elsner (2006), you consider three coastal regions including the Gulf Coast, Florida, and the East Coast (Fig. 6.2). Regions are large enough to capture enough hurricanes, but not too large as to include many non-coastal strikes. Here you use hourly position and intensity data described in Chapter 6. For each hurricane, you note its wind speed maximum within each region. If the maximum wind exceeds 33 m s^{-1}, then you count it as a hurricane for the region. A tropical cyclone that affects more than one region at hurricane intensity is counted in each region. Because of this, the sum of the regional counts is larger than the total count.

Begin by loading **annual.RData**. These data were assembled in Chapter 6. Subset the data for years starting with 1866.

```
> load("annual.RData")
> dat = subset(annual, Year >= 1866)
```

The covariate Southern Oscillation Index (SOI) data begins in 1866. Next, extract all hurricane counts for the Gulf coast, Florida, and East coast regions.

```
> cts = dat[, c("G.1", "F.1", "E.1")]
```

11.1.1 Cluster Detection

You start by comparing the observed with the expected number of years for the two groups of hurricane counts. The groups include years with no hurricanes and years with three or more. The expected number is from a Poisson distribution with a constant rate. The idea is that for regions that show a cluster of hurricanes, the observed number of years with no hurricanes and years with three or more hurricanes should be greater than the corresponding expected number. Said another way, a Poisson model with a hurricane rate estimated from counts over all years in regions with clustering will underestimate the number of years with no hurricanes and years with many hurricanes.

For example, you find the observed number of years without a Florida hurricane and the number of years with more than two hurricanes by typing

```
> obs = table(cut(cts$F.1,
+   breaks=c(-.5, .5, 2.5, Inf)))
> obs
(-0.5,0.5]  (0.5,2.5]  (2.5,Inf]
        70         62         13
```

And the expected numbers for these three groups by typing

```
> n = length(cts$F.1)
> mu = mean(cts$F.1)
> exp = n * diff(ppois(c(-Inf, 0, 2, Inf), lambda=mu))
> exp
[1] 61.66 75.27  8.07
```

The observed and expected counts for the three regions are given in Table 11.1. In the Gulf and East coast regions, the observed number of years are relatively close to the expected number of years in each of the groups. By contrast, in the Florida region, you see the observed number of years exceeds the expected number of years for the no-hurricane and the three-or-more hurricanes groups.

Table 11.1 Observed (O) and expected (E) number of hurricane years by count groups.

Region	$O(=0)$	$E(=0)$	$O(\geq 3)$	$E(\geq 3)$
Gulf Coast	63	66.1	7	6.6
Florida	70	61.7	13	8.1
East Coast	74	72.3	7	4.9

Table 11.2 Observed versus expected statistics. The Pearson and χ^2 test statistics along with the corresponding p-values are given for each coastal region.

Region	*Pearson*	*p-value*	χ^2	*p-value*
Gulf Coast	135	0.6858	0.264	0.894
Florida	187	0.0092	6.475	0.039
East Coast	150	0.3440	1.172	0.557

The difference between the observed and expected numbers is used to assess the statistical significance of the clustering. This is done using Pearson residuals and the χ^2 statistic. The Pearson residual is the difference between the observed count and expected rate divided by the square root of the variance. The p-value is evidence in support of the null hypothesis of no clustering.

For example, to obtain the χ^2 statistic, type

```
> xis = sum((obs - exp)^2 / exp)
> xis
[1] 6.48
```

The p-value as evidence in support of the null hypothesis is given by

```
> pchisq(q=xis, df=2, lower.tail=FALSE)
[1] 0.0393
```

where `df` is the degrees of freedom equal to the number of groups minus one. The χ^2 and Pearson statistics for the three regions are shown in Table 11.2. The p-values for the Gulf and East coasts are greater than 0.05 indicating little support for the cluster hypothesis. By contrast, the p-value for the Florida region is 0.009 using the Pearson residuals and 0.037 using the χ^2 statistic. These values provide evidence that the hurricane occurrences in Florida are clustered in time.

11.1.2 Conditional Counts

What might be causing this clustering? The extra variation in annual hurricane counts might be due to variation in hurricane rates. You examine this possibility with a

Poisson regression model (see Chapter 7). The model includes an index for the NAO and the SOI after Elsner and Jagger (2006).

This is a generalized linear model (GLM) approach using the Poisson family that includes a logarithmic link function for the rate. The annual hurricane count model is

$$H_i \sim \text{dpois}(\lambda_i) \tag{11.1}$$
$$\log(\lambda_i) = \beta_0 + \beta_{\text{soi}}\,\text{SOI}_i + \beta_{\text{nao}}\,\text{NAO}_i$$

where H_i is the hurricane count in year i simulated ($\sim$) from a Poisson distribution with a rate λ_i that depends on the year i. The logarithm of the rate depends in a linear way on the SOI and NAO covariates. The code to fit the model, determine the expected count, and table the observed versus expected counts for each region is given below.

```
> pfts = list(G.1 = glm(G.1 ~ nao + soi,
+   family="poisson", data=dat),
+   F.1 = glm(F.1 ~ nao + soi, family="poisson",
+   data=dat),
+   E.1 = glm(E.1 ~ nao + soi, family="poisson",
+   data=dat))
> prsp = sapply(pfts, fitted, type="response")
> rt = regionTable(cts, prsp, df=3)
```

The count model provides an expected number of hurricanes in each year. The expected is compared to the observed as before. Results indicate that clustering is somewhat ameliorated by conditioning the rates on the covariates. In particular, the Pearson residual reduces to 172.4 with an increase in the corresponding p-value to 0.042. However, the p-value remains near 0.05 indicating the conditional model, while an improvement, fails to capture all the extra (beyond Poisson) variation in Florida hurricane counts.

11.1.3 Cluster Model

Having found evidence that Florida hurricanes arrive in clusters next you model this process. In the simplest case, you assume the following. Each cluster has either one or two hurricanes and the annual cluster count is described by a Poisson distribution with a rate r. Note the difference. Earlier, you assumed that each hurricane was independent and the annual hurricane count followed a Poisson distribution. Furthermore, now you let p be the probability that a cluster will have two hurricanes.

Formally, your model can be expressed as follows. Let N be the number of clusters in a given year and $X_i, i = 1, \ldots, N$ be the number of hurricanes in each cluster minus one. Then, the number of hurricanes in a given year is $H = N + \sum_{i=1}^{N} X_i$. Conditional on N, $M = \sum_{i=1}^{N} X_i$ is described by a binomial distribution since the X_is are independent Bernoulli variables and p is constant. That is, $H = N + M$, where the annual number of clusters N has a Poisson distribution with cluster rate r, and M

has a binomial distribution with proportion p and size N. Here the binomial distribution describes the number of occurrences of at least one hurricane in a sequence of N independent years, with each year having a probability p of observing at least one hurricane.

In summary, your cluster model has the following properties:

1. The expected number of hurricanes $E(H) = r(1+p)$.
2. The variance of H is given by

$$\begin{aligned}\text{var}(H) &= E(\text{var}(H|N)) + \text{var}(E(H|N))\\ &= E(N(p(1-p))) + \text{var}((1+p)N)\\ &= rp(1-p) + r(1+p)(1+p) = r(1+3p)\end{aligned}$$

3. The dispersion of H is given by $\text{var}(H)/E(H) = \phi = (1+3p)/(1+p)$, which is independent of cluster rate. Solving for p, you find $p = (\phi - 1)/(3-\phi)$.
4. The probability mass function for the number of hurricanes, H, is,

$$P(H = k|r,p) = \sum_{i=0}^{\lfloor i/2 \rfloor} \text{dpois}(k-i,r)\ \text{dbinom}(i,k-i,p); k = 0,1,\ldots$$

$$\begin{aligned}P(H = 0|r,p) &= e^{-r}\text{dpois}(k-i,r)\\ &= e^{-r}\frac{r^{k-i}}{(k-i)!}\text{dbinom}(i,k-i,p)\\ &= \frac{k-i}{i}p^{i}(1-p)^{k-2i}\end{aligned}$$

5. The model has two parameters r and p. A better parameterization is to use $\lambda = r(1+p)$ with p to separate the hurricane frequency from the cluster probability. The parameters do not need to be fixed and can be functions of the covariates.
6. When $p = 0$, H is Poisson, and when $p = 1$, $H/2$ is Poisson, the dispersion is two, and the probability that H is even is 1.

You need a way to estimate r and p.

11.1.4 Parameter Estimation

Your goal is a hurricane count distribution for the Florida region. For that, you need to estimate the annual cluster rate (r) and the probability (p) that the cluster size is two. Continuing with the GLM approach, you separately estimate the annual hurricane frequency, λ, and the annual cluster rate r. The ratio of these two parameters minus one is an estimate of the probability p.

This is reasonable if p does not vary much, since the annual hurricane count variance is proportional to the expected hurricane count (i.e., $\text{var}(H) = r(1+3p) \propto r \propto E(H)$). You estimated the parameters of the annual count model using Poisson regression, which assumes that the variance of the count is proportional to the expected count. Thus under the assumption that p is constant, Poisson regression can be used to estimate λ in the cluster model.

You regress the logarithm of the link function for the cluster rate onto the predictors NAO and SOI. The model is given by

$$
\begin{aligned}
N_i &\sim \text{dpois}(r_i) \qquad (11.2)\\
\log(r_i) &= \alpha_0 + \alpha_1\,\text{SOI}_i + \alpha_2\,\text{NAO}_i + \varepsilon_i
\end{aligned}
$$

The parameters of this annual cluster count model cannot be estimated directly, since the observed hurricane count does not furnish information about the number of clusters.

Consider the observed set of annual Florida hurricane counts. Since the annual frequency is quite small, the majority of years have either no hurricanes or a single hurricane. You can create a "reduced" data set by using an indictor of whether or not there was at least one hurricane. Formally, let $I_i = I(H_i > 0) = I(N_i > 0)$), then I is an indicator of the occurrence of a hurricane cluster for each year. You assume that I has a binomial distribution with size parameter of one and a proportion equal to π. This leads to a logistic regression model (see Chapter 7) for I.

Note that since $\exp(-r)$ is the probability of no clusters, the probability of a cluster π is $1-\exp(-r)$. Thus the cluster rate is $r = -\log(1-\pi)$. If you use a logarithmic link function on r, then $\log(r) = \log(-\log(1-\pi)) = \text{cloglog}(\pi)$, where cloglog is the complementary log—log function. Thus, you model I using the cloglog function to obtain r.

Your cluster model is a combination of two models, one for the counts and another for the clusters. Formally, it is given by

$$
\begin{aligned}
I_i &\sim \text{dbern}(\pi_i) \qquad (11.3)\\
\text{cloglog}(\pi_i) &= \alpha_0 + \alpha_1\,\text{SOI}_i + \alpha_2\,\text{NAO}_i
\end{aligned}
$$

where `dbern` is the Bernoulli distribution with mean π. The covariates are the same as those used in the cluster count model. Given these equations, you have the following relationships for r and p.

$$\log(\hat{r}(1+\hat{p})) = \hat{\beta}_0 + \hat{\beta}_1\,\text{SOI} + \hat{\beta}_2\,\text{NAO} \qquad (11.4)$$

$$\log(\hat{r}) = \hat{\alpha}_0 + \hat{\alpha}_1\,\text{SOI} + \hat{\alpha}_2\,\text{NAO} \qquad (11.5)$$

By subtracting the coefficients in Eq. 11.5 for the annual cluster count model from the those in Eq. 11.4 for the annual hurricane count model, you have a regression model for the probabilities given by

$$\log(1+\hat{p}) = \hat{\beta}_0 - \hat{\alpha}_0 + (\hat{\beta}_1 - \hat{\alpha}_1)\,\text{SOI} + (\hat{\beta}_2 - \hat{\alpha}_2)\,\text{NAO} \qquad (11.6)$$

11.1.5 Model Diagnostics

You start by comparing fitted values from the count model with fitted values from the cluster model. Let H_i be the hurricane count in year i and $\hat{\lambda}_i$ and $\hat{r}_i$ be the fitted annual count and cluster rates, respectively. Then let τ_0 be a test statistic given by

$$\tau_0 = \frac{1}{n}\sum_{i=1}^{n}(H_i - \hat{r}_i) = \frac{1}{n}\sum_{i=1}^{n}(\hat{\lambda}_i - \hat{r}_i). \tag{11.7}$$

The value of τ_0 is greater than one if there is clustering. You test the significance of τ_0 by generating random samples of length n from a Poisson distribution with rate λ_i and computing τ_j for $j = 1, \ldots, N$, where N is the number of samples. A test of the null hypothesis that $\tau_0 \leq 0$ is the proportion of simulated τ's that are at least as large as τ_0.

You do this with the `testfits` function in the **correlationfuns.R** package by specifying the model formula, data, and number of random samples.

```
> source("correlationfuns.R")
> tfF = testfits(F.1 ~ nao + soi, data=dat, N=1000)
> tfF$test; tfF$testpvalue
[1] 0.104
[1] 0.026
```

For Florida hurricanes, the test statistic τ_0 has a value of 0.104 indicating some difference in count and cluster rates. The proportion of 1,000 simulated τs that are at least as large as this is 0.026 providing sufficient evidence to reject the no-cluster hypothesis. Repeating the simulation using Gulf Coast hurricanes

```
> tfG = testfits(G.1 ~ nao + soi, data=dat, N=1000)
> tfG$test; tfG$testpvalue
[1] -0.0617
[1] 0.787
```

you find that, in contrast to Florida, there is little evidence against the no-cluster hypothesis. The results are consistent with what you found previously using the χ^2 statistic.

A linear regression through the origin of the fitted count rate on the cluster rate under the assumption that p is constant yields an estimate for $1 + p$. You plot the annual count and cluster rates and draw the regression line using the `plotfits` function.

```
> par(mfrow=c(1, 2), pty="s")
> ptfF = plotfits(tfF)
> mtext("a", side=3, line=1, adj=0, cex=1.1)
> ptfG = plotfits(tfG)
> mtext("b", side=3, line=1, adj=0, cex=1.1)
```

The regressions are shown in Figure 11.1 for Florida and the Gulf coast. The black line is the $y = x$ axis, and cluster and hurricane rates align along this axis if there is no clustering. The red line is the regression of the fitted hurricane rate onto the fitted cluster rate with the intercept set to zero. The slope is an estimate of $1 + p$.

Table 11.3 Coefficients of the count rate model.

	Estimate	*Std. Error*	*z Value*	*Pr(>|z|)*
(Intercept)	−0.27	0.11	−2.55	0.01
nao	−0.23	0.09	−2.50	0.01
soi	0.06	0.03	1.86	0.06

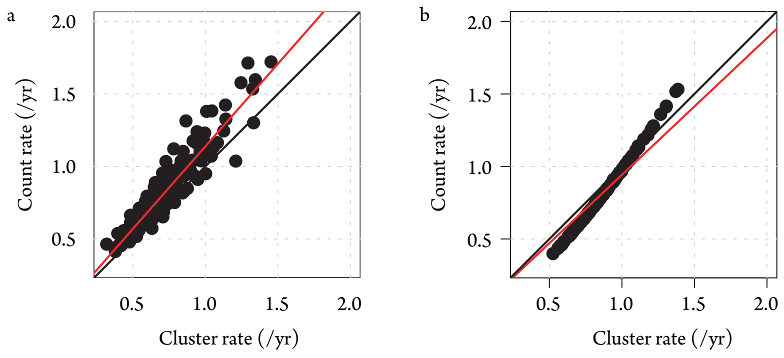

Figure 11.1 Count versus cluster rates for (a) Florida and (b) Gulf coast.

The regression slopes are printed by typing,

```
> coefficients(ptfF); coefficients(ptfG)
rate
1.14
 rate
0.942
```

The slope is 1.14 for the Florida region giving 0.14 as an estimate for p (probability that the cluster will have two hurricanes). The regression slope is 0.94 for the Gulf coast region, which you interpret as a lack of evidence for hurricane clusters.

Your focus is now on Florida hurricanes only. You continue by looking at the coefficients from both models. Type

```
> summary(tfF$fits$poisson)$coef
> summary(tfF$fits$binomial)$coef
```

The output coefficient tables (Tables 11.3 and 11.4) show that the NAO and SOI covariates are significant in the hurricane count model, but only the NAO is significant in the cluster rate model.

The difference in coefficient values from the two models is an estimate of $\log(1+p)$, where again p is the probability that a cluster will have two hurricanes. The difference in the NAO coefficient is 0.043 and the difference in the SOI coefficient is 0.035 indicating that the NAO increases the probability of clustering more

Table 11.4 Coefficients of the cluster rate model.

	Estimate	*Std. Error*	*z Value*	*Pr(>|z|)*
(Intercept)	−0.42	0.13	−3.11	0.00
nao	−0.27	0.12	−2.23	0.03
soi	0.02	0.04	0.52	0.60

than ENSO. Lower values of the NAO lead to a larger rate increase for the Poisson model relative to the binomial model.

11.1.6 Forecasts

It is interesting to compare forecasts of the distribution of Florida hurricanes using your Poisson and cluster models. Here you set $p = .138$ for the cluster model. You can use the same two-component formulation for your Poisson model by setting $p = 0$.

You prepare the data using the `lambdapclust` function as follows:

```
> ctsF = cts[, "F.1", drop=FALSE]
> pars = lambdapclust(prsp[, "F.1", drop=FALSE],
+   p=.138)
> ny = nrow(ctsF)
> h = 0:5
```

Next, you compute the expected number of years with `h` hurricanes from the cluster and Poisson models and tabulate the observed number of years. You combine them in a data object.

```
> eCl = sapply(h, function(x)
+    sum(do.call('dclust',
+    c(x=list(rep(x, ny)), pars))))
> ePo = sapply(0:5, function(x)
+    sum(dpois(x=rep(x,ny),
+    lambda=prsp[, "F.1"])))
> o = as.numeric(table(ctsF))
> dat = rbind(o, eCl, ePo)
> names(dat) = 0:5
```

Finally, you plot the observed versus the expected from your cluster and Poisson models using a bar plot where the bars are plotted side by side.

```
> barplot(dat, ylab="Number of Years",
+    xlab="Number of Florida Hurricanes",
+    names.arg=c(0:5),
+    col=c("black", "red", "blue"),
```

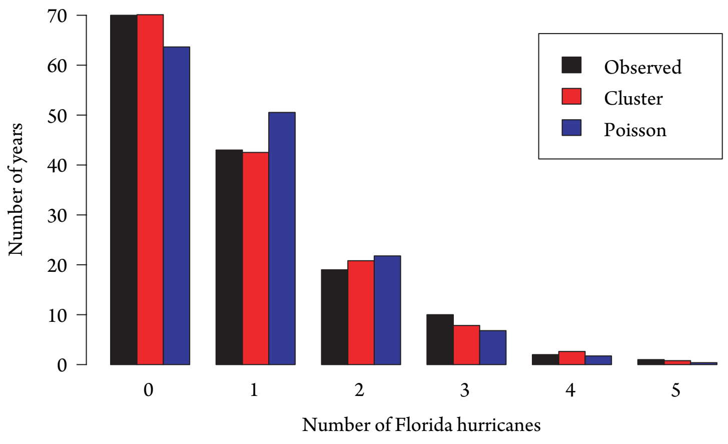

Figure 11.2 Observed versus expected number of Florida hurricane years.

```
+     legend=c("Observed", "Cluster", "Poisson"),
+     beside=TRUE)
```

Results are shown in Figure 11.2. The expected numbers are based on your cluster model ($p = 0.137$) and on your Poisson model ($p = 0$). The cluster model fits the observed counts better than does the Poisson model particularly at the low and high count years.

Florida had hurricanes in only 2 of the 11 years from 2000 through 2010. But these 2 years featured seven hurricanes. Seasonal forecast models that predict U.S. hurricane activity assume a Poisson distribution. You show here that this assumption applied to Florida hurricanes leads to a forecast that underpredicts both the number of years without hurricanes and the number of years with three or more hurricanes (Jagger and Elsner, 2012).

11.2 SPATIAL CLUSTERS

Is there a tendency for hurricanes to cluster in space? More specifically, given that a hurricane originates at a particular location, is it more (or less) likely that another hurricane will form in the same vicinity? This question can be answered using spatial point pattern analysis. Here you consider models for analyzing and modeling events across space.

We begin with some definitions. An *event* is the occurrence of some phenomenon of interest. For example, an event can be a hurricane's origin or its lifetime maximum intensity. An *event location* is the spatial location of the event. For example, the latitude and longitude of the maximum intensity event. A *point* is any location where an event could occur. A *spatial point pattern* is a collection of events and event locations together with spatial domain. The *spatial domain* is defined as the region of interest over which events occur. For example, the North Atlantic basin is the spatial domain for hurricanes.

Complete spatial randomness (CSR) defines a situation where an event is equally likely to occur at any location within the study area regardless of the locations of other events. A spatial point pattern is said to be clustered if given an event at some location it is more likely than random that another event will occur nearby. Regularity is the opposite; given an event, if it is less likely than random that another event will occur nearby, then the spatial point pattern is regular.

A *realization* is a collection of spatial point patterns generated under a spatial point process model. To illustrate, consider four point patterns each consisting of events inside the unit plane. You generate the event locations and plot them by typing

```
> par(mfrow=c(2, 2), mex=.9, pty="s")
> for(i in 1:4){
+ u = runif(n=30, min=0, max=1)
+ v = runif(n=30, min=0, max=1)
+ plot(u, v, pch=19, xlim=c(0, 1), ylim=c(0, 1))
+ }
```

The pattern of events are shown in Figure 11.3. The plots which show events under CSR illustrate that some amount of clustering occurs by chance.

The **spatstat** package (Baddeley and Turner, 2005) contains a large number of functions for analyzing and modeling point pattern data. To make the functions available and obtain a citation, type

```
> require(spatstat)
> citation(package="spatstat")
```

CSR lies on a continuum between clustered and regular spatial point patterns. You use simulation functions in **spatstat** to compare the CSR plots in Figure 11.3 with regular and clustered patterns. Examples are shown in Figure 11.4. Here you see two

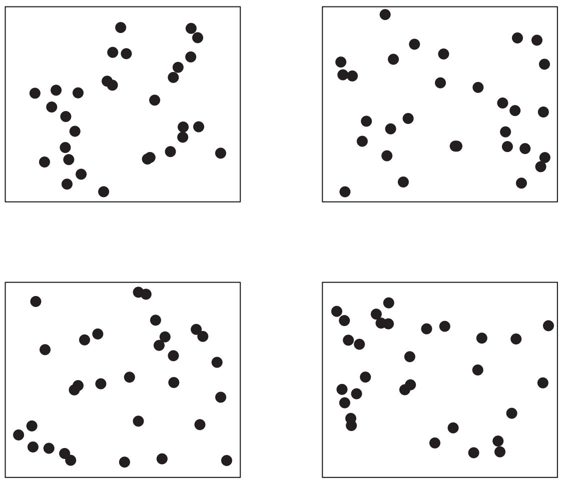
Figure 11.3 Point patterns exhibiting complete spatial randomness.

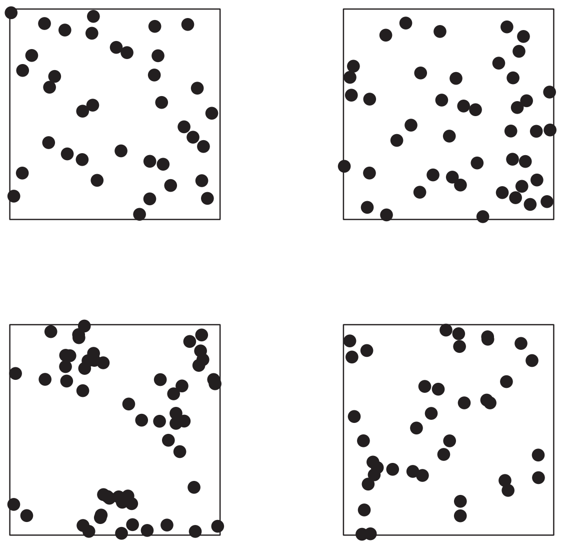
Figure 11.4 Regular (top) and clustered (bottom) point patterns.

realizations of patterns more regular than CSR (top row), and two realizations of patterns more clustered than CSR (bottom row). The simulations are made using a spatial point pattern model. Spatial scale also plays a role. A set of events can indicate a regular spatial pattern on one scale but a clustered pattern on anther.

11.2.1 Point Processes

Given a set of spatial events (spatial point pattern), your goal is to assess evidence for clustering (or regularity). Spatial cluster detection methods are based on statistical models for spatial point processes. The random variable in these models is the event locations. Here we provide some definitions useful for understanding point process models. Details on the underlying theory are available in Ripley (1981), Cressie (1993), and Diggle (2003).

A process generating events is *stationary* when it is invariant to translation across the spatial domain. This means that the relationship between two events depends only on their positions relative to one another and not on the event locations themselves. This relative position refers to (lag) distance and orientation between the events. A process is *isotropic* when orientation does not matter. Recall these same concepts applied to the variogram models used in Chapter 9.

Given a single realization, the assumptions of stationarity and isotropy allow for replication. Two pairs of events in the same realization of a stationary process that are separated by the same distance should have the same relatedness. This allows you to use relatedness information from one part of the domain as a replicate for relatedness in another part of the domain. The assumptions of stationarity and isotropy are the starting point but can be relaxed later on.

The Poisson distribution is used to define a model for CSR. A spatial point process is said to be *homogeneous* Poisson under the following two criteria:

- The number of events occurring within a finite region A is a random variable following a Poisson distribution with mean $\lambda \times A$ for some positive constant λ, with $|A|$ denoting the area of A.
- Given the total number of events N, their locations are an independent random sample of points where each point is equally likely to be picked as an event location.

The first criterion is about the density of the spatial process. For a given domain, it answers the question, "how many events?" It is the number of events divided by the domain area. The density is an estimate of the rate parameter of the Poisson distribution.[1] The second criterion is about homogeneity. Events are scattered throughout the domain and are not clustered or regular.

It is instructive to consider how you would create a homogeneous Poisson point pattern. The procedure follows from the definition. Step 1: Generate the number of events using a Poisson distribution with mean equal to λ. Step 2: Place the events inside the domain using uniform distributions for the spatial coordinates.

For example, let area of the domain be one and the density of events be 200, then type

```
> lambda = 200
> N = rpois(1, lambda)
> x = runif(N); y=runif(N)
> plot(x, y, pch=19)
```

Note that if your domain is not a rectangle, you can circumscribe it within a rectangle, then reject events sampled from inside the rectangle that fall outside the domain. This is called "rejection sampling."

By definition, these point patterns are CSR. However, as noted earlier, you are typically in the opposite position. You observe a set of events and want to know if they are regular or clustered. Your null hypothesis is CSR and you need a test statistic that will guide your inference. CSR models are easy to construct, so you can use Monte Carlo methods.

In some cases, the homogeneous assumption is too restrictive. Consider hurricane genesis as an event process. Event locations may be random, but the probability of an event is higher away from the equator. The constant risk hypothesis of the homogeneous Poisson point pattern model requires a generalization to include a spatially varying density function.

To do this, you define the density $\lambda(s)$, where s denotes spatial points. This is called an *inhomogeneous* Poisson model, and it is analogous to the universal kriging model used on field data (see Chapter 9). Inhomogeneity as defined by a spatially varying density implies nonstationarity as the number of events depends on location.

[1] In the spatial statistics, this is often called the *intensity*. Here we use the term *density* instead so as not to confuse the spatial rate parameter with hurricane strength.

11.2.2 Spatial Density

Density is a first-order property of a spatial point pattern. That is, the density function estimates the mean number of events at any point in your domain. Events are independent of one another, but event clusters might appear because of the varying density.

Given an observed set of events, how do you estimate $\lambda(s)$? One approach is to use kernel densities. Consider again the set of hurricanes over the period 1944–2000 that were designated tropical only and baroclinically enhanced (Chapter 7). Input these data and create a spatial points data frame of the genesis locations by typing

```
> bh = read.csv("bi.csv", header=TRUE)
> require(sp)
> coordinates(bh) = c("FirstLon", "FirstLat")
> ll = "+proj=longlat +ellps=WGS84"
> proj4string(bh) = CRS(ll)
```

Next convert the geographic coordinates using the Lambert conformal conic projection true at parallels 10 and 40°N and a center longitude of 60°W. First save the reference system as a CRS object, then use the `spTransform` function from the **rgdal** package.

```
> lcc = "+proj=lcc +lat_1=40 +lat_2=10 +lon_0=-60"
> require(rgdal)
> bht = spTransform(bh, CRS(lcc))
```

The spatial distance unit is meters. Use the `map` (**maps**) and the `map2SpatialLines` (**maptools**) to obtain country borders by typing

```
> require(maps)
> require(maptools)
> brd = map("world", xlim=c(-100, -30), ylim=c(10, 48),
+   plot=FALSE)
> brd_ll = map2SpatialLines(brd, proj4string=CRS(ll))
> brd_lcc = spTransform(brd_ll, CRS(lcc))
```

You use the same coordinate transformation on the map borders as you do on the hurricane locations.

Next you convert your S4 class objects (`bh` and `clp`) into S3 class objects for use with the functions in the **spatstat** package.

```
> require(spatstat)
> bhp = as.ppp(bht)
> clpp = as.psp(brd_lcc)
```

The spatial point pattern object `bhp` contains marks. Mark are attributes attached to the events. By default the marks are the columns from the original data that are not used for location. You are interested only in the hurricane type (either tropical only or baroclinic), so you reset the marks accordingly by typing

```
> marks(bhp) = bht$Type == 0
```

You summarize the object with the `summary` method. The summary includes an average density over the spatial domain. The density is per unit area. Your length unit is meter from the Lambert projection, so your density is per square meter. You retrieve the average density in units of per (1,000 km^2) by typing

```
> summary(bhp)$intensity * 1e+12
[1] 11.4
```

Thus, on average, each point in your spatial domain has slightly more than 10 hurricane origins per 1,000 km^2. This is the mean spatial density. Some areas have more or less than the mean so you would like an estimate of $\lambda(s)$.

You do this using the `density` method, which computes a kernel smoothed density function from your point pattern object. By default, the kernel is Gaussian with a bandwidth equal to the standard deviation of the isotropic kernel. The default output is a pixel image containing local density estimates. You again convert the density values to units of 1,000 km^2.

```
> den = density(bhp)
> den$v = den$v * 1e+12
```

Finally you use the plot method to plot the densities, the country borders, and the event locations.

```
> plot(den)
> plot(unmark(clpp), add=TRUE)
> plot(unmark(bhp), pch=19, cex=.3, add=TRUE)
```

Event density maps split by tropical-only and baroclinic hurricane genesis are shown in Figure 11.5. Here we converted the *im* object to a *SpatialGridDataFrame* object and used the `spplot` method. Regions of the central Atlantic extending westward through the Caribbean into the southern Gulf of Mexico have the greatest tropical-only density generally exceeding 10 hurricane origins per (1000 km^2). By contrast, regions along the eastern coast of the United States extending eastward to Bermuda have the greatest baroclinic density. The amount of smoothing is determined by the bandwidth and the type of kernel.

Densities at the genesis locations are made using the argument `at="points"` in the `density` call. Also the number of events in grid boxes across the spatial domain are obtained using the `quadratcount` or `pixellate` functions. For example, type

```
> plot(quadratcount(bhp))
> plot(pixellate(bhp, dimyx=5))
```

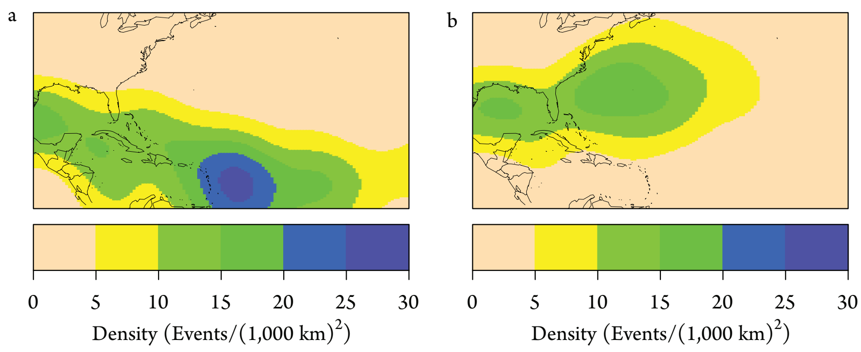
a
b
0
5
10
15
20
25
30
Density (Events/(1,000 km)^2)
0
5
10
15
20
25
30
Density (Events/(1,000 km)^2)

Figure 11.5 Genesis density for (a) tropical-only and (b) baroclinic hurricanes.

11.2.3 Second-Order Properties

The density function describes the rate (mean) of hurricane genesis events locally. Second-order functions describe the variability of events. Ripley's K function is an example. It is defined as

$$\hat{K}(s) = \lambda^{-1} n^{-1} \sum_{i \neq j} I(d_{ij} < s), \tag{11.8}$$

where d_{ij} is the Euclidean distance between the ith and jth events in a data set of n events, and λ is the average density of events, estimated as $n/|A|$, where $|A|$ is the area of the domain. I is an indicator that equals 1 if the distance is less than or equal to s.

The definition assumes stationarity and isotropy for the point pattern and implies that under CSR if the events are homogeneous, $\hat{K}(s)$ should be approximately equal to πs^2. For point patterns more regular than CSR, you expect fewer events within distance s of a randomly chosen event than under CSR, so $\hat{K}(s) < \pi s^2$. For point patterns more clustered than CSR, you expect more events within a given distance than under CSR or $\hat{K}(s) > \pi s^2$.

The function `Kest` is used to compute $\hat{K}(s)$. Here you save the results by typing

```
> k = Kest(bhp)
```

The function takes an object of class `ppp` and computes $\hat{K}(s)$ using a formula that corrects for edge effects to reduce bias arising from counting hurricanes near the borders of your spatial domain (Ripley, 1991; Baddeley et al., 2000).

The K function is defined such that $\lambda\hat{K}(s)$ equals the expected number of additional hurricanes within a distance s of any other hurricane. You plot the expected number as a function of separation distance by typing

```
> lam = summary(bhp)$intensity
> m2km = .001
> plot(k$r * m2km, k$iso * lam, type="l", lwd=2,
+       xlab="Lag Distance (s) [km]",
+       ylab="Avg Number of Nearby Hurricanes")
```

You first save the value of λ and a conversion factor to go from meters to kilometers. The `iso` column refers to the K function computed using the isotropic correction for regular polygon domains.

The `Kest` function also returns theoretical values under the hypothesis of CSR. You add these to your plot by typing

```
> lines(k$r * m2km, k$theo * lam, col="red", lwd=2)
```

The empirical curve lies above the theoretical curve at all lag distances. For instance, at a lag distance of 471 km, there are on average about 16 additional hurricanes nearby. This compares with an expected number of eight additional hurricanes. This indicates that for a given hurricane, there are more nearby hurricanes than you would expect by chance under the assumption of CSR. This is indicative of clustering.

But as you have seen earlier, the clustering is related to the inhomogeneous distribution of hurricane events across the basin. So this result is not particularly useful. Instead you use the `Kinhom` function to compute a generalization of the K function for inhomogeneous point patterns (Baddeley et al, 2000). The results are shown in Figure 11.6. Black curves are the empirical estimates and red curves are the theoretical. The empirical curve is much closer to the inhomogeneous theoretical curve.

There appears to be some clustering of hurricane genesis at short distances and regularity at larger distances. The regularity at large distance is likely due to land. The analysis can be improved by masking land areas.

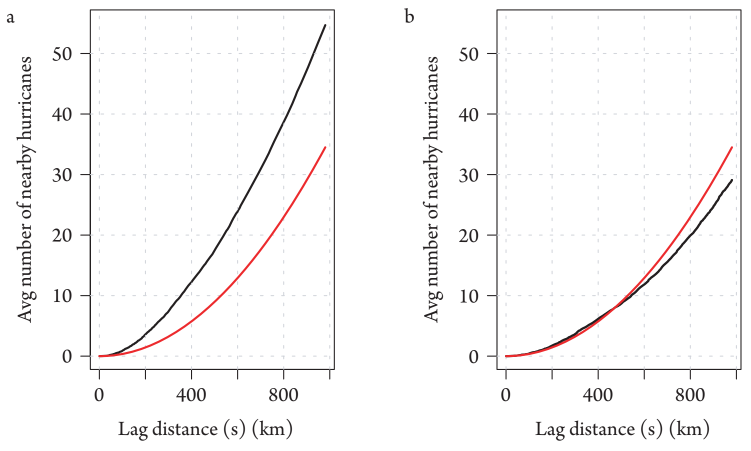

Figure 11.6 2nd-order genesis statistics. (a) Ripley's K and (b) generalization of K.

11.2.4 Models

You can fit a parametric model to your point pattern. This allows you to predict the occurrence of hurricane events given covariates and the degree of clustering. The **spatstat** package contains options for fitting point pattern models with the `ppm` function.

Models can include spatial trend, dependence on covariates, and event interactions. Models are fit using the method of maximum pseudo-likelihood, or using an approximate maximum likelihood method. Model syntax is standard R. Typically, the null model is homogeneous Poisson.

11.3 FEATURE CLUSTERS

In the first two sections of this chapter, you examined clustering in time and geographic space. It is also possible to cluster in feature space. Indeed, this is the best known of the cluster methods. Feature clustering is called cluster analysis.

Cluster analysis searches for groups in feature (attribute) data. Objects belonging to the same cluster have similar features. Objects belonging to different clusters have dissimilar features. In two or three dimensions, clusters can be visualized. In more than three, analytic assistance is helpful.

Note the difference from the two earlier cluster methods. With those methods your initial goal was cluster detection with the final goal a model to improve prediction. Here, your goal is descriptive and cluster analysis is a data-reduction technique. You start with the assumption that your data can be grouped based on feature similarity.

Cluster analysis methods fall into two distinct categories: partition and hierarchical. Partition clustering divides your data into a prespecified number of groups. The number of groups is usually chosen by trial. An index of cluster quality can make it easier for you to choose an optimal number.

Hierarchical clustering creates increasing or decreasing sets of clustered groups from your data. Agglomerative hierarchical starts with each object forming its own cluster. It then successively merges clusters until a single large cluster remains. Divisive hierarchical is the opposite. It starts with a single cluster that includes all objects and successively splits the clusters into smaller groups.

To cluster in feature space, you need to

1. create a dissimilarity matrix from a set of objects, and
2. group the objects based on the values of the dissimilarity matrix.

11.3.1 Dissimilarity and Distance

The dissimilarity between two objects measures how different they are. The larger the dissimilarity, the greater the difference. Objects are considered vectors in attribute (feature) space, where vector length equals the number of data set attributes.

Consider, for instance, a data set with three attributes and four objects (Table 11.5). Object 1 is a vector consisting of the triplet $(x_{1,1}, x_{1,2}, x_{1,3})$, object 2

Table 11.5 Attributes and objects. A data set ready for cluster analysis.

	Feature 1	*Feature 2*	*Feature 3*
Object 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
Object 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
Object 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$
Object 4	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$

is a vector consisting of the triplet $(x_{2,1}, x_{2,2}, x_{2,3})$, and so on. Then the elements of a dissimilarity matrix are pairwise distances between the objects in feature space. As an example, an object might be a hurricane track with features that include genesis location, lifetime maximum intensity, and lifetime maximum intensification.

Although distance is an actual metric, the dissimilarity function need not be. The minimum requirements for a dissimilarity measure d are as follows:

- $d_{i,i} = 0$
- $d_{i,j} \geq 0$
- $d_{i,j} = d_{j,i}$.

The following axioms of a proper metric may be included but are not necessary:

- $d_{i,k} \leq d_{i,j} + d_{j,k}$ triangle inequality
- $d_{i,j} = 0$ implies $i = j$.

Before clustering, you arrange your data set as a $n \times p$ data matrix, where n is the number of rows (one for each object) and p is the number of columns (one for each attribute variable).

How you compute dissimilarity depends on your attribute variables. If the variables are numeric, you use Euclidean or Manhattan distance given as

$$d_{i,j}^{E} = \sqrt{\sum_{f=1}^{p} (x_{i,f} - x_{j,f})^2} \tag{11.9}$$

$$d_{i,j}^{M} = \sum_{f=1}^{p} |x_{i,f} - x_{j,f}| \tag{11.10}$$

The summation is over the number of attributes (features). With hierarchical clustering, the maximum distance norm, given by

$$d_{i,j}^{\max} = \max(|x_{i,f} - x_{j,f}|, f = 1, \ldots, p) \tag{11.11}$$

is sometimes used.

Measurement units on your feature variables influence the relative distances feature space which, in turn, will affect your clusters. Features with high variance will have the largest impact. If all features are deemed equally important to your grouping, then the data need to be standardized. You can do this with the `scale` function, whose default method centers and scales the columns of your data matrix.

The `dist` function computes the distance between objects using a variety of methods including Euclidean (default), Manhattan, and maximum. Create two feature vectors each having five values and plot them with object labels in feature space.

```
> x1 = c(2, 1, -3, -2, -3)
> x2 = c(1, 2, -1, -2, -2)
> plot(x1, x2, xlab="Feature 1", ylab="Feature 2")
> text(x1, x2, labels=1:5, pos=c(1, rep(4, 4)))
```

From the plotted points, you can group the two features into two clusters by eye. There is an obvious distance separation between the clusters. To compute pairwise Euclidean distances between the five objects, type

```
> d = dist(cbind(x1, x2))
> d
     1    2    3    4
2 1.41
3 5.39 5.00
4 5.00 5.00 1.41
5 5.83 5.66 1.00 1.00
```

The result is a vector of distances, but printed as a table with the object numbers listed as row and column names. The values are the pairwise distances so, for instance, the distance between object 1 and object 2 is 1.41 units. You can see two distinct clusters of distances (dissimilarities) those less than or equal to 1.41 and those greater than or equal to 5. Objects 1 and 5 are the most dissimilar followed by objects 2 and 5. On the other hand, objects 3 and 5 and 4 and 5 are the most similar. You can change the default Euclidean distance to the Manhattan distance by including `method="man"` in the `dist` function.

If the feature vectors contain values that are not continuous numeric (e.g., factors, ordered, binary) or if there is a mixture of data types (one feature is numeric and the other is a factor, for instance), then dissimilarities need to be computed differently. The **cluster** package contains functions for cluster analysis including `daisy` for dissimilarity matrix calculations, which by default uses Euclidean distance as the dissimilarity metric.

To test, type

```
> require(cluster)
> d = daisy(cbind(x1, x2))
> d
Dissimilarities :
     1    2    3    4
2 1.41
3 5.39 5.00
```

```
4 5.00 5.00 1.41
5 5.83 5.66 1.00 1.00

Metric :  euclidean
Number of objects : 5
```

Note, the values that make up the dissimilarity matrix are the same as the earlier distance values, but additional information is saved including the metric used and the total number of objects in your data set. The function contains a logical flag called `stand` that when set to true standardizes the feature vectors before calculating dissimilarities. If some of the features are not numeric, the function computes a generalized coefficient of dissimilarity (Gower, 1971).

11.3.2 K-Means Clustering

Partition clustering requires you to specify the number of clusters beforehand. This number is denoted *k*. The algorithm allocates each object in your data frame to one and only one of the *k* clusters.

The *k*-means method is the most popular. Membership of an object is determined by its distance from the centroid of each cluster. The centroid is the multidimensional version of the mean. The method alternates between calculating the centroids based on the current cluster members and reassigning objects to clusters based on the new centroids.

For example, in deciding which of the two clusters an object belongs, the method computes the dissimilarity between the object and the centroid of cluster one and between the object and the centroid of cluster two. It then assigns the object to the cluster with the smallest dissimilarity and recomputes the centroid with this new object included. An object already in this cluster might now have greater dissimilarity due to the reposition of the centroid, in which case it gets reassigned. Assignments and reassignments continue in this way until all objects have a membership.

The `kmeans` function from the base **stat** package performs *k*-means clustering. By default, it uses the algorithm of Hartigan and Wong (1979). The first argument is the data frame (not the dissimilarity matrix) and the second is the number of clusters (centers). To perform a *k*-means cluster analysis on the example data above, type

```
> dat = cbind(x1, x2)
> ca = kmeans(dat, centers=2)
> ca
K-means clustering with 2 clusters of sizes 3, 2

Cluster means:
     x1    x2
1 -2.67 -1.67
2  1.50  1.50
```

```
Clustering vector:
[1] 2 2 1 1 1

Within cluster sum of squares by cluster:
[1] 1.33 1.00
 (between_SS / total_SS =  93.4 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"
```

Initial centroids are chosen at random so it is recommended to rerun the algorithm a few times to see if it arrives at the same groupings. While the algorithm minimizes within-cluster variance, it does not ensure that there is a global minimum variance across all clusters.

The first bit of output gives the number of clusters (input) and the size of the clusters that results. Here cluster 1 has two members and cluster 2 has three. The next bit of output are the cluster centroids. There are two features labeled `x1` and `x2`. The rows list the centroids for clusters 1 and 2 as vectors in this two-dimensional feature space. The centroid of the first cluster is $(-2.67, -1.67)$ and the centroid of the second cluster is $(1.5, 1.5)$.

The centroid is the feature average using all objects in the cluster. You can see this by adding the centroids to your plot.

```
> points(ca$centers, pch=8, cex=2)
> text(ca$centers, pos=c(4, 1),
+   labels=c("Cluster 2", "Cluster 1"))
```

The next bit of output tags each object with cluster membership. Here you see that the first two objects belong to cluster 2 and the next three belong to cluster 1. The cluster number keeps track of distinct clusters, but the numerical order is irrelevant.

The within-cluster sum of squares is the sum of the distances between each object and the cluster centroid for which it is a member. From your earlier plot you can see that the centroids (stars) minimize the within-cluster distances while maximizing the between-cluster distances.

The function `pam` uses medoids rather than centroids. A medoid is a multidimensional center based on medians. The method accepts a dissimilarity matrix, tends to be more robust (converges to the same result), and provides for additional graphical displays from the **cluster** package.

11.3.3 Track Clusters

Here your interest is to group hurricanes according to common track features. These features include location of hurricane origin, location of lifetime maximum intensity, location of final hurricane intensity, and lifetime maximum intensity. The three location features are further subdivided into latitude and longitude making a total of seven attribute (feature) variables. Note that location is treated here as an attribute rather than as a spatial coordinate.

You first create a data frame containing only the attributes you wish to cluster. Here you use the cyclones since 1950:

```
> load("best.use.RData")
> best = subset(best.use, Yr >= 1950)
```

You then split the data frame into separate cyclone tracks using `Sid` with each element in the list as a separate track. You are interested in identifying features associated with each track.

```
> best.split = split(best, best$Sid)
```

You then assemble a data frame using only the attributes to be clustered.

```
> best.c = t(sapply(best.split, function(x){
+   x1 = unlist(x[1, c("lon", "lat")])
+   x2 = unlist(x[nrow(x), c("lon", "lat")])
+   x3 = max(x$WmaxS)
+   x4 = unlist(x[rev(which(x3 == x$WmaxS))[1],
+     c("lon", "lat")])
+   return(c(x1, x2, x3, x4))
+ }))
> best.c = data.frame(best.c)
> colnames(best.c) = c("FirstLon", "FirstLat",
+   "LastLon", "LastLat", "WmaxS", "MaxLon", "MaxLat")
```

The data frame contains seven features from 667 cyclones.

Before clustering, you check the feature variances by applying the `var` function on the columns of the data frame using the `sapply` function.

```
> sapply(best.c, var)
FirstLon FirstLat  LastLon  LastLat    WmaxS
   499.2     57.0    710.4    164.8    870.7
  MaxLon   MaxLat
   356.7     75.9
```

The variances range from a minimum of 57 degrees squared for the latitude of origin feature to a maximum of 870.7 knots squared for the maximum intensity feature. Because of this large range, you scale the features so that each will have the same influence on the clustering.

```
> best.cs = scale(best.c)
> m = attr(best.cs, "scaled:center")
> s = attr(best.cs, "scaled:scale")
```

The function `scale` centers and scales the columns of your numeric data frame. The center and scale values are saved as attributes in the new data frame. Here you save them to rescale the centroids after the cluster analysis.

You perform a *k*-means cluster analysis setting the number of clusters to three by typing

```
> k = 3
> ct = kmeans(best.cs, centers=k)
> summary(ct)
             Length Class  Mode
cluster      667    -none- numeric
centers       21    -none- numeric
totss          1    -none- numeric
withinss       3    -none- numeric
tot.withinss   1    -none- numeric
betweenss      1    -none- numeric
size           3    -none- numeric
```

The output is a list of length 7 containing the cluster vector, cluster means (centroids), the total sum of squares, the within-cluster sum of squares by cluster, the total within sum of squares, the between sum of squares, and the size of each cluster.

Your cluster means are scaled so they are not readily meaningful. The cluster vector gives the membership of each hurricane in order as they appear in your data set. The ratio of the between sum of squares to the total sum of squares is 44.7 percent. This ratio will increase with the number of clusters, but at the expense of having clusters that may not be interpretable. With four clusters, the increase in this ratio is smaller than the increase going from two to three clusters. So you are content with your three-cluster analysis.

11.3.4 Track Plots

Since six of the seven features are spatial coordinates it is tempting to plot the centroids on a map and connect them with a track line. This would be misleading since the combination of attributes is not a spatial feature. Instead, you plot examples of cyclones that resemble each cluster.

First, you add cluster membership and distance to your original data frame, then split the data by cluster member.

```
> ctrs = ct$center[ct$cluster, ]
> cln = ct$cluster
> dist = rowMeans((best.cs - ctrs)^2)
> id = 1:length(dist)
> best.c = data.frame(best.c, id=id, dist=dist,
+   cln=cln)
> best.c.split = split(best.c, best.c$cln)
```

Next you subset your cluster data based on the tracks that come closest to the cluster centroids. This closeness occurs is in feature space that includes not only latitude and longitude but also intensity. Here you choose nine tracks for each centroid.

```
> te = 9
> bestid = unlist(lapply(best.c.split, function(x)
+   x$id[order(x$dist)[1:te]]))
> cinfo = subset(best.c, id %in% bestid)
```

Finally, you plot the tracks on a map. This requires a few steps to make the plot easier to read. Begin by setting the bounding box using latitude and longitude of your cluster data and renaming your objects.

```
> cyclones = best.split
> uselat = range(unlist(lapply(cyclones[cinfo$id],
+   function(x) x$lat)))
> uselon = range(unlist(lapply(cyclones[cinfo$id],
+   function(x) x$lon)))
```

Next, order the clusters by number and distance and set the colors for plotting. Use the `brewer.pal` function in the **RColorBrewer** package Neuwirth (2011). You use a single hue sequential color ramp that allows you to highlight the tracks that are closest to the centroids of the feature clusters.

```
> cinfo = cinfo[order(cinfo$cln, cinfo$dist), ]
> require(RColorBrewer)
> blues = brewer.pal(te, "Blues")
> greens = brewer.pal(te, "Greens")
> reds = brewer.pal(te, "Reds")
> cinfo$colors = c(rev(blues), rev(greens), rev(reds))
```

Next, reorder the tracks for plotting them as a weave with the tracks farthest from the centroids plotted first.

```
> cinfo = cinfo[order(-cinfo$dist, cinfo$cln), ]
```

Finally, plot the tracks and add world and country borders. Results are shown in Figure 11.7. Track color is based on attribute proximity to the cluster centroid using a color saturation that decreases with distance.

```
> plot(uselon, uselat, type="n", xaxt="n", yaxt="n",
+   ylab="", xlab="")
> for(i in 1:nrow(cinfo)){
+   cid = cinfo$id[i]
+   cyclone = cyclones[[cid]]
```

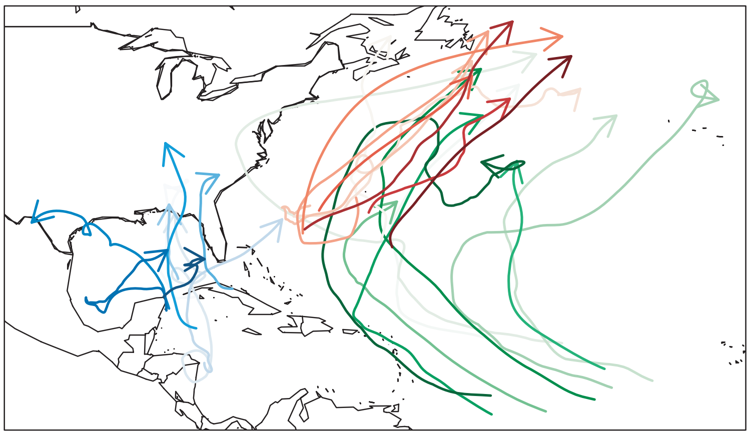

Figure 11.7 Tracks by cluster membership.

```
+   lines(cyclone$lon, cyclone$lat, lwd=2,
+     col=cinfo$colors[i])
+ }
> require(maps)
> map("world", add=TRUE)
> map("usa", add=TRUE)
```

The analysis splits the cyclones into a group over the Gulf of Mexico, a group at high latitudes, and a group that begins at low latitudes but ends at high latitudes generally east of the United States. Some of the cyclones that begin at high latitude are baroclinic (see Chapter 7).

Cluster membership can change depending on the initial random centroids particularly for tracks that are farthest from a centroid. The two- and three-cluster tracks are the most stable. The approach can be extended to include other track features including velocity, acceleration, and curvature representing more dimensions of the space and time behavior of hurricanes.

If you want to make inferences about future hurricane activity. Model-based clustering, where the observations are assumed to be quantified by a finite mixture of probability distributions, is an attractive alternative to cluster analysis. In the end, it is worthwhile to keep in mind the advice of Bill Venables and Brian Ripley. You should not assume cluster analysis is the best way to discover interesting groups. Indeed, visualization methods are often more effective in this task.

This chapter showed you how to detect, model, and analyze clusters in hurricane data. We began by showing you how to detect and model the arrival of hurricanes along the coast. We then showed you how to detect and analyze the presence of spatial clusters in hurricane genesis locations. We also looked at the first- and second-order statistical properties of hurricane genesis. Finally, we showed you how to apply cluster analysis to hurricane track features and map representative tracks.

In the next chapter, we look at several applications of Bayesian models.