1 Spatial Point Pattern Data

The tornado data are organized in a slightly different way than the hurdat2 dataset.

Each row of the dataset is represents one tornado, whereas each tropical cyclone in the hurdat dataset could span multiple rows.

Some important fields in the tornado dataset are:

- date - date in yyyy-mm-dd format
- time - time in HH:MM:SS format (usually central standard time)
- tz - time zone
- f - F scale or EF scale (after 2007)
- inj - injuries
- fat - fatalities
- loss - property loss value
- closs - crop loss value
- slat, slon - starting coordinates
- elat, elon - ending coordinates
- len - track length (could be verified)
- wid - track width

Statisticians call this type of dataset a point pattern dataset. In a point pattern dataset, the spatial locations represent locations of events (much like the hurdat data).

This is in contrast to "geostatistical" data, where the locations are more like metadata. The locations in geostatistical data are the place where someone decided to take a measurement of some other variable, such as air temperature. In Geostatistical data, the response is that measurement, with the idea that the measurement could have been taken at any location. Point pattern data, on the other hand, represent locations of events.

This dataset would sometimes be called a "marked" point pattern dataset, because, along with the event locations, other information is collected, such as the strength of the event (EF scale), the number of injuries, crop losses, etc. Most point pattern datasets are marked point pattern datasets.

Just like with hurricanes, tornados are more or less likely to occur at different locations and times. Often it is of interest to use the data to produce a map of where tornados are likely to occur. These maps may vary with time of year or time of day, which is one of the suggest analyses.

For notational purposes, let's label the storms $i = 1, \ldots, n$, where $n = 71{,}813$.

Define $s_i = (slon_i, slat_i)$ to be the starting coordinates. The observed locations contain information about where tornados are likely to occur, but in their raw form, they are usually not useful for making maps unless they are smoothed out.

The main logical justification for smoothing is the following: If we see a tornado at location s_i, that provides information that tornados might occur close to location s_i but doesn't provide information

that tornados might occur far from s_i. If we have an arbitrary location x, the closer x is to s_i, the more information that s_i provides about the propensity for storms at location x.

Given that reasoning, he is how people often "smooth out" the raw locations. Define a grid of locations $x_1, \ldots, x_G$, where G is the number of grid points. And calculate the smoothed function

$$\widehat{\lambda}(x_j) = \frac{1}{n} \sum_{i=1}^{n} K(x_j - s_i) \tag{1}$$

Where K is called a kernel function. A common choice is a Gaussian function

$$K(x_j - s_i) = \frac{1}{C} \exp(-\|x_j - s_i\|/\alpha) \tag{2}$$

where α defines the width of the kernel, and C is a normalization factor ensuring that the kernel sums to 1.

There is an art to selecting α, which should depend on how many data points you have, but also on your sense of what an event at s_i means for the propensity of events near s_i.

We can think of the kernel smoother as counting up the number of nearby events, but each event s_i only "counts" the amount $K(x_j - s_i)$ for location x_j, where the amount is larger for events that are closer to x_j.

Kernel density estimation is implemented in the `spatstat` R package, and I'm sure in other packages as well.

Spatial Distances on the Globe

As pointed out by one of the presentations, a natural distance measure when working in latitude longitude coordinates is the great circle distance, which is the shortest distance you can travel across the surface of a sphere between two points.

The `fields` package in R has a function called `rdist.earth` for calculating great circle distance from latitudes and longitudes.

A brief aside on distances on spheres. Another natural distance measure is the Euclidean distance in 3-dimensional space. This is the distance between two points on a sphere if you are allowed to travel in a straight line through the Earth.

There is a 1-1 correspondence between great circle distance and euclidean distance, in the sense that if I asked you for all of the events within a great circle distance of 1000 km, this is equivalent to asking for all of the events within a Euclidean distance of $2 \cdot 6371 \sin(1000/(2 \cdot 6371)) = 998.9738$ km.

These two distances are essentially equal for small distances since $\sin(x) \approx x$ for small x.

Spatial Poisson Process Models

For the hurricanes, we primarily viewed the hurricane events as indexed by time, but we can also index them by spatial location. The same is true of the tornado data.

Let's for now ignore the time component and talk about models for spatial events.

The heterogeneous Poisson process is easy to define for spatial events:

Y is a random function that takes in a region A as its argument, and returns the number of events that occur within that region.

The Poisson process model for Y has the following two assumptions:

$$Y(A) \sim \text{Poisson}\Big(\int_A \lambda(x)dx\Big)$$

$Y(A_1) \perp\!\!\!\perp Y(A_2)$ if regions A_1 and A_2 have no common locations.

As in the heterogeneous Poisson process for hurricane event times, the heterogeneous spatial Poisson process has a rate function λ, which can be interpreted as the expected number of events per unit area.

We used $\widehat{\lambda}(x)$ for the kernel density estimate, because we can view $\widehat{\lambda}(x)$ as an estimate of the rate function $\lambda(x)$.

If you have a $\lambda(x)$, simulating from heterogeneous spatial poisson process model is analogous to how we did it for the temporal process process model.

Two Steps. Let Ω denote the entire region over which we wish to simulate

1. Simulate $J \sim \text{Poisson}(\int_\Omega \lambda(x)dx)$
2. Sample $x_1, \ldots, x_J$ independently from a density proportional to $\lambda(x)$ on Ω

This requires us to be able to integrate $\lambda(x)$ over Ω and to simulate the locations. The integration can usually be approximated numerically (though this tends to be slow), and there are various methods for sampling the locations, such as rejection sampling.

Estimating Log-Linear Heterogeneous Poisson Process Models

Coming up with parametric models for $\lambda(x)$ is challenging. One option is to model it as a linear function of some covariate, such as latitude, elevation, or some other topographic variable.

Let $w(x)$ be the measurable covariate. This doesn't work:

$$\lambda(x) = b_0 + b_1 w(x)$$

because it doesn't constrain $\lambda(x)$ to be positive, so people usually do this:

$$\lambda(x) = \exp(b_0 + b_1 w(x)) \quad \Longleftrightarrow \quad \log \lambda(x) = b_0 + b_1 w(x)$$

Then the task is to estimate two parameters b_0 and b_1, instead of trying to estimate an entire function $\lambda(x)$.

As usual, we turn to the loglikelihood function to help us estimate parameters. To construct the loglikelihood, we use the simulation algorithm to guide us. Let $x_1, \ldots, x_n$ be the observations, and define

$$I(b_0, b_1, \Omega) = \int_\Omega \exp(b_0 + b_1 w(x))dx$$

The likelihood is

$$L(b_0, b_1) = \frac{I(b_0, b_1, \Omega)^n \exp(-I(b_0, b_1, \Omega))}{n!} \prod_{i=1}^{n} \frac{\exp(b_0 + b_1 w(x_i))}{I(b_0, b_1, \Omega)}$$

You get some cancellation of the integrals, which gives

$$L(b_0, b_1) = \frac{\exp(-I(b_0, b_1, \Omega))}{n!} \prod_{i=1}^{n} \exp(b_0 + b_1 w(x_i))$$

To interpret the likelihood, you can see that the product term is larger when $\exp(b_0 + b_1 w(x_i))$ is larger, but the integral term gets smaller when the integral of the rate function increases, so there is a balance. You can't just pick b_0 and b_1 to make the rate function really big because you pay a price for that with the other term.

In practice, you have to numerically integrate the intensity, which is possible, but this is often slow, so people convert the location data into counts within a partition of the domain (quadrat counts).

Here's how that works.

Define a set of grid boxes that covers the domain. Call these boxes $A_1, \ldots, A_G$. Calculate $y_1, \ldots, y_G$ as $y_i = y(A_i)$, that is, the number of events within grid box A_i. And define $|A_i|$ to be the area of A_i, and assume that all areas are equal to the same size C

Under the Poisson process models, $y_1, \ldots, y_n$ can be modeled as independent Poisson random variables with expectation $I(b_0, b_1, A_i) = \int_{A_i} \lambda(x)dx$, so the likelihood is

$$L(b_0, b_1) = \prod_{i=1}^{G} \frac{I(b_0, b_1, A_i)^{y_i} \exp(-I(b_0, b_1, A_i))}{y_i!}$$

Since it's a product, we can separate things out

$$\begin{aligned}
L(b_0, b_1) &= \prod_{i=1}^{G} \exp(-I(b_0, b_1, A_i)) \prod_{i=1}^{G} I(b_0, b_1, A_i)^{y_i} \prod_{i=1}^{G} \frac{1}{y_i} \\
&= \exp\Big(- \sum_{i=1}^{G} I(b_0, b_1, A_i) \Big) \prod_{i=1}^{G} I(b_0, b_1, A_i)^{y_i} \prod_{i=1}^{G} \frac{1}{y_i} \\
&= \exp(-I(b_0, b_1, \Omega)) \prod_{\text{all } i \text{ such that } y_i = 1} I(b_0, b_1, A_i) \\
&\approx \exp(-I(b_0, b_1, \Omega)) \prod_{i=1}^{n} \exp(b_0 + b_1 w(x_i)) C
\end{aligned}$$

The switch from producting up to G to producting up to n is allowed because when y_i is 0, raising I to the power zero gives 1. So they are essentially the same likelihood, albeit with different constants, but the constants don't affect where the maximum with respect to b_0 and b_1 occurs.

To fit this model in R, all you really need to do is convert to quadrat counts, stored in a vector `y`, and also store the covariate in a vector `w`, and then use the glm function, as in

```
m1 <- glm(y ~ w, family = "Poisson" )
```

This will use the following likelihood:

$$L(b_0, b_1) = \prod_{i=1}^{G} \frac{(\exp(b_0 + b_1 w(A_i))|A_i|)^{y_i} \exp(-\exp(b_0 + b_1 w(A_i))|A_i|)}{y_i!}$$

where $w(A_i)$ is however you have defined the value of the covariate for the ith grid box. Usually people pick the value of w at the center of grid box A_i.

However, the above code only works when the areas of the grid boxes (quadrats) are all the same. If the areas differ, you can tell the glm function to use an offset

```
m1 <- glm(y ~ w, family = poisson(link=logit), offset = log(areas) )
```

This is because the log of the mean of the poisson distribution is

$$\log \mu_i = b_0 + b_1 w(A_i) + \log |A_i| \tag{3}$$

The offset essentially adds another covariate but does not assign a coefficient to it, which is the right thing to do here.

Software

The most widely used (I think) R package for analyzing point pattern data is the spatstat package. You can find documentation on their website:

https://spatstat.org/