Analysis and Prediction of Call For Service Data

Warren L. G. Koontz

1 Introduction

The National Institute of Justice (NIJ) recently sponsored a *Real-Time Crime Forecasting Challenge* that "seeks to harness the advances in data science to address the challenges of crime and justice." Participants were challenged to identify a forecast area within the city of Portland, OR where certain types of crime are most likely to happen. NIJ provided historical crime data for the Portland police district for the period January 1, 2013 through December 31, 2016. Participants could use this data and any other available data to make their predictions. Entries were judged based on measurements of actual crime data for a three month period beginning March 2017. For a complete description of the challenge, see `https://nij.gov/funding/Pages/fy16-crime-forecasting-challenge.aspx`.

I submitted an entry to the competition as a "small team/business" (consisting only of myself). This paper describes how I used tools provided by MATLAB to analyze the historical data and determine an area within the Portland police district that seemed most likely to have the highest rate of a particular crime.

NIJ announced the results in early August of 2017 and I was among the winners.

2 Data Provided by NIJ

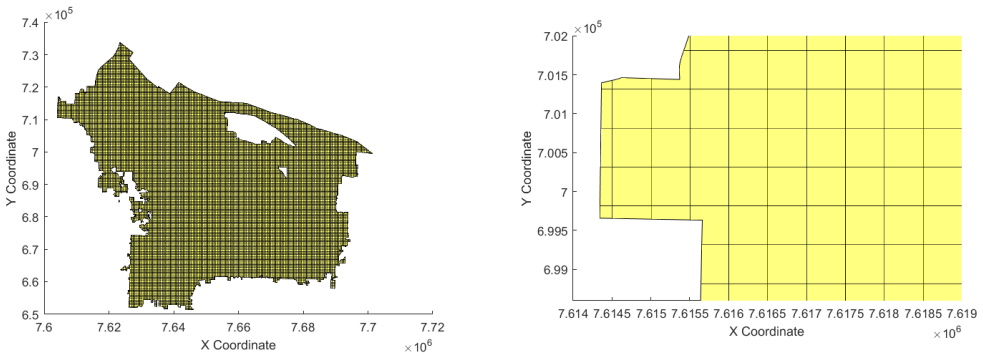
The primary data provided by NIJ are records of each *call for service* (CFS) received by the Portland police district during the period 1/1/2013 to 12/31/2016. Each CFS record includes a location (x,y coordinate), crime category and subcategory, a date, and some other information. The data are contained in *shape files*[1], which are designed to be used by mapping software. The MATLAB Mapping Toolbox provides functions `shaperead` and `shapewrite` to read and write these files. Within the MATLAB environment, the data is stored in a geographic data structure. To read a shape file into MATLAB, write

```
S=shaperead(filename);
```

where `filename` is the name of the shape file. The function `shaperead` loads the CFS data into an $N \times 1$ structure `S`, where N is the number of CFS. Each element of the structure contains a number of fields, but only the following fields are of interest here:

`X` – x coordinate of CFS (in ft.)

`Y` – y coordinate of CFS (in ft.)



(a) Map of Portland OR. (b) Close-up of Map of Portland OR.

Figure 1: Display of Shape File Map Data.

`CATEGORY` – Category of CFS

The function `shaperead` also supports selective reading of the shape file. To limit `S` to CFS classified as burglaries, write

```
B=shaperead(filename,'Selector',{@(v) strcmp(v,'BURGLARY'),...
    'CATEGORY'});
```

The NIJ also provided a shape file containing map data for the city of Portland. Whereas the CFS shape files associate each CFS with a point, the map file is a collection of polygons, where each polygon is a small section of the city. The `X` and `Y` fields are vectors of the vertices of the polygons. Most of the polygons are rectangles of the same size, but many along the perimeter of the city are more complex. The area of each polygon is provided in a field named `area`. Figure 1a displays the map data and Figure 1b provides a closer view of the western edge. These figures were produced using the MATLAB function `mapshow`, which accepts a geometric data structure as the input argument.

3 Evaluation Criterion

NIJ judged entries based on two criteria: a prediction accuracy index (PAI) and a prediction efficiency index (PEI). The PAI is defined as

$$\text{PAI} = \frac{n/N}{a/A} \tag{1}$$

where n equals the number of crimes that occur in the forecast area, N equals the total number of crimes, a equals the area of the forecast area, and A equals the area of the entire study area. The forecast area was required to satisfy $0.25 \leq a \leq 0.75$, where area is measured in square miles. The PEI is defined as

$$\text{PEI} = \frac{n}{n^*} \tag{2}$$

where n^* is the largest number of crimes that occur in an area of size a within Portland. As described below, my approach is mostly driven by the PAI criterion.

4 Analysis Approach

Assume that the locations of the CFS are samples from a bivariate distribution with a probability density function (PDF) $p(x, y)$. We can use historical CFS data to obtain an estimate, $\hat{p}(x, y)$, of this PDF. We then use this estimate to determine the forecast area.

The kernel smoothing method, originally developed by E. Parzen[2], remains a popular means for providing an estimate of a PDF based on samples drawn from the PDF. For bivariate data, the kernel smoothing estimate of the PDF is given by[3]

$$\hat{p}(x, y) = \frac{1}{N} \sum_{n=1}^{N} k(x - x_n, h_x) k(y - y_n, h_y) \tag{3}$$

where $k(u, h)$ is the *kernel function*, which can vary considerably, but generally satisfies

$$\begin{aligned} k(u, h) &\geq 0 \\ k(-u, h) &= k(u, h) \\ \int_{-\infty}^{\infty} k(u, h) du &= 1 \end{aligned} \tag{4}$$

The parameter h is called the *bandwidth* of the kernel and determines how broad or narrow the kernel function is. For example, in the Gaussian kernel given by

$$k(u, h) = \frac{1}{\sqrt{2\pi} h} e^{-u^2/2h^2} \tag{5}$$

h resembles the standard deviation of a Gaussian random variable.

For univariate and bivariate data, the kernel smooting estimate of the PDF can be determined using the MATLAB function `ksdensity`. By default, `ksdensity` uses a Gaussian kernel and computes the bandwidth internally, The most straightforward application of `ksdensity` is to simply write

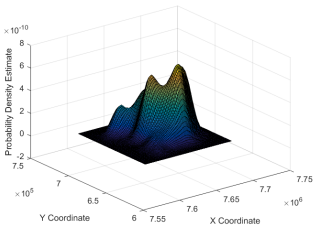
```
ksdensity(X)
```

where `X` is a $N \times 2$ matrix of the sample points and N is the sample size. Called in this way, `ksdensity` produces a plot of the PDF estimate. Thus, given the structure `B` containing burglary data, we can plot an estimate of the burglary PDF as follows:

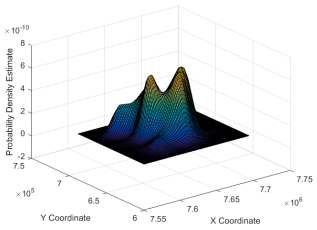
```
ksdensity([[B.X]' [B.Y]'])
```

Several such plots are shown in Figure 2, which also includes plots for motor vehicle theft.

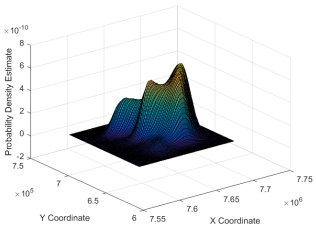
By default, `ksdensity` computes $\hat{p}(x, y)$ for points on a mesh grid covering the ranges of x and y in the sample. However, it is also possible for `ksdensity` to compute $\hat{p}(x, y)$



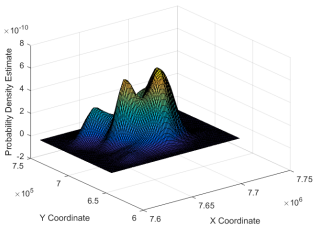
(a) PDF Estimate: 2013 Burglary.



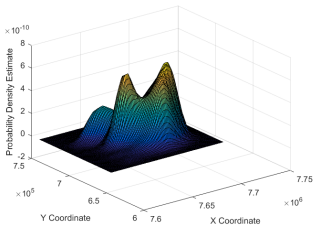
(b) PDF Estimate: 2014 Burglary.



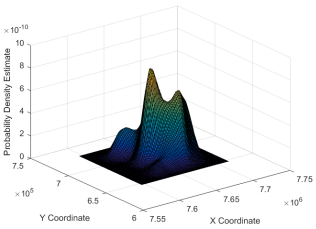
(c) PDF Estimate: 2015 Burglary.



(d) PDF Estimate: 2013 MV Theft.



(e) PDF Estimate: 2014 MV Theft.



(f) PDF Estimate: 2015 MV Theft.

Figure 2: PDF Estimates for Crime Data.

for specified values of x and y. Our approach is to compute the PDF estimate for a representative point in each polygon in the map data. We can compute the representative points using the following MATLAB script:

```
S=shaperead(mapData);
M=length(S);
pts=zeros(M,2);
for m=1:M
    pts(m,1)=nanmean(S(m).X);
    pts(m,2)=nanmean(S(m).Y);
end
```

The representative points are stored in an $M \times 2$ array `pts`. Each row of `pts` contains an x, y pair computed from the average of the `X,Y` values for the vertices of the corresponding polygon. The function `nanmean` computes the average value ignoring the `NaN` (not a number) values that show up in some of the more complex polygons.

Having determined the representative points, we can compute the PDF estimate for each point as

```
pdf=ksdensity([[B.X]' [B.Y]'],pts);
```

The result `pdf` is a $M \times 1$ array of PDF values and `pdf(m)` is the PDF estimate for the m^{th} polygon in the map data.

Let I be a subset of $\{1, 2, \ldots, M\}$ indicating those polygons that comprise the forecast

area (the "hot spots"). Then the theoretical PAI is

$$\text{PAI} = A\frac{\sum_{n\in I} p_n a_n}{\sum_{n\in I} a_n} \tag{6}$$

where p_n represents `pdf(n)` and a_n represents `S(n).area`. Note that the numerator in Equation (6)

$$P = \sum_{n\in I} p_n a_n \tag{7}$$

is the probability that a given CFS occurs in the forecast area. So the object of the game is to find the subset I that maximizes the PAI subject to the constraint on the size of the forecast area. A reasonable, but not necessarily optimal solution can be obtained by sorting the elements of `pdf` in descending order and creating the forecast area from the polygons corresponding to the largest values in the `pdf` array. We start by using the MATLAB function `sort` to order the list of polygons.

```
[~,I]=sort(pdf,'descending');
```

Called this way, `sort` returns a vector `I` of indices such that `I(k)` is the index of the k^{th} largest value of `pdf`. We can now use the following code to identify the hot spots and compute the theoretical PAI.

```
P=0;
a=0;
amax=0.75*(5280^2);
for m=1:M
    k=I(m);
    a=a+S(k).area;
    if a > amax
        a=a-S(k).area;
        break
    end
    P=P+pdf(k)*S(k).area;
    S(k).hotspot=1;
end
PAI=A*P/a;
```

This code adds polygons to the forecast area in the order specified in `I` until the size of the forecast area reaches the target `amax`, which in this case is the maximum allowed area. It also flags the polygons that comprise the forecast area and computes the numerator sum in Equation (6), which is in turn used to calculate the theoretical PAI. The field `hotspot` is initialized to zero for all of the polygons.

5 Implementation and Test

I implemented the analysis approach of Section 4 as a MATLAB script. The input to the script consists of three shape files

Table 1: Motor Vehicle Theft Data – Maximum Area

Total area	147.703130 sq mi
Forecast area	0.744301 sq mi (0.5%, 83 polygons)
Theoretical PAI	3.152938
Total CFS in training set	2487
Predicted CFS in forecast area	40
Actual CFS in forecast area	59 (2.4%)
Training set PAI	4.707794
Training set n*	349
Training set PEI	0.169054
Total CFS in test set	287
Predicted CFS in forecast area	5
Actual CFS in forecast area	4 (1.4%)
Test set PAI	2.765790
Test set n*	103
Test set PEI	0.038835

- CFS data for calendar year 2016 – the *training set*
- CFS data for January 2017 – the *test set*
- Map data for the Portland police district

The script begins by reading the map data, determining the representative point for each polygon, initializing the `hotspot` fields to zero, and computing the total area of the police district for verification. Then it reads in the training set, computes the PDF estimate, and uses the PDF estimate to identify the hot spots and the theoretical PAI (the theoretical PEI is 1.0). At this point, the map data, which now includes the populated `hotspot` fields, is saved using `shapewrite` for submission to the competition.

The script continues, however, in order to calculate further results using the training and test sets:

- Predicted CFS in forecast area – the product of P given by Equation (7) and the total number of CFS in the set
- Actual CFS in forecast area – the total number of CFS in polygons flagged as hot spots (n).[1]
- Measured PAI and PEI – based on Equations (6) and (2)

The quantity n^* used to calculate the PEI is obtained by counting the number of CFS in each polygon (i.e., creating a histogram), sorting the polygons in descending order of CFS, and summing the top M, where M is the number of polygons in the forecast area.

[1] the MATLAB function `inpolygon` is helpful here

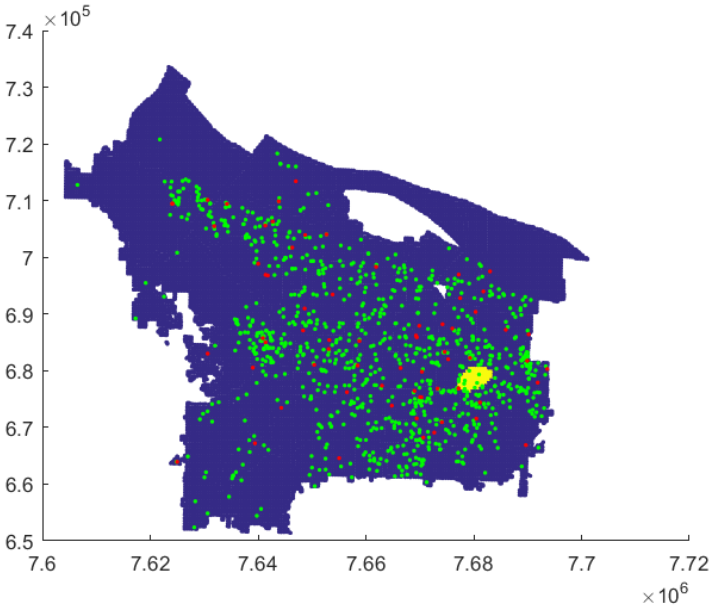


Figure 3: forecast Area and CFS Locations

Table 1 lists the results of running the script for motor vehicle theft with `amax` set to the maximum value (0.75 sq mi). The total area value matches the value given by the NIJ. The forecast area is just under 0.75 sq mi (0.5% of the total area) and includes 83 polygons. The theoretical PAI is about 3.15 and, based on the associated probability given by Equation (7), 40 of the 2487 CFS in the training set should occur in the forecast area. The actual number is 59, so that the calculated PAI is greater than the theoretical value. On the other hand, 349 CFS occur in an area equal in size to the forecast area, so that the measured PEI is rather low. As is often the case, the test set performance is poorer with a PAI of about 2.77 and a PEI of about 0.04.

An analysis of Equation (6) suggests that the theoretical PAI increases as the forecast area decreases. In fact, with `amax` set to the minimum (0.25 sq mi), the theoretical PAI is 2.60598. However, the training set PAI drops to 1.209145 and the test set PAI drops to 0! Thus, with a small number of training or test samples plus a bit of bad luck, the actual PAI deviates considerably from the theoretical value. For this reason, I used the maximum area for my submission

The script also produces a map, as shown in Figure 3, That displays the forecast area (yellow), the training set CFS (green), and the test set CFS (red).

Given the low value of PEI, even for the training set, one might suppose that it would be better to use the training set histogram to select the hot spots. Indeed, the training set PAI and PEI would be 27.9 and 1.0. I decided, however, to stay with the PDF based approach, which seems less susceptible to overfitting and tends to produce a contiguous forecast area.

6 Entry Submission and Results

The competition was divided by entrant type, crime category, time period, and measurement criterion. The entrant types were student, small team/business, and large business; the crime categories were burglary, street crime, theft of auto, and all crimes; the time periods were 1 week, 2 weeks, 1 month, 2 months, and 3 months; the measurement criteria were PAI and PEI as defined earlier. I entered as a small team and limited my entry to burglary and theft of auto. I used the CFS data for 2016 to determine a forecast area for each of the two crime categories and used the same forecast area for each time period. Although I based my approach on maximizing the PAI criterion, all entries were evaluated using both criteria. Thus my entry covered twenty combinations of crime category, time period, and criterion.

According to the results announced by the NIJ in early August 2017, I tied with seven other entrants for the following combination:

- Entrant type: small team/business
- Crime category: burglary
- Time period: 1 week
- Criterion: PEI

References

[1] I. Esr, "Esri shapefile technical description," September 1998.

[2] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 09 1962. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177704472

[3] A. W. Bowman and A. Azzalini, *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*. OUP Oxford, 1997, vol. 18.