Scout Model

Brad Lindblad, MBA © 2019 - Confidential & Proprietary. Civitas Secure, LLC

April 29, 2019

Abstract

A model to ingest and transform incident-level crime data into normalized and scaled format. Data is gathered from various APIs at which point various statistical methods transform the data set into a format that is conducive to a 'Log-normal-distributed' user experience.

Contents

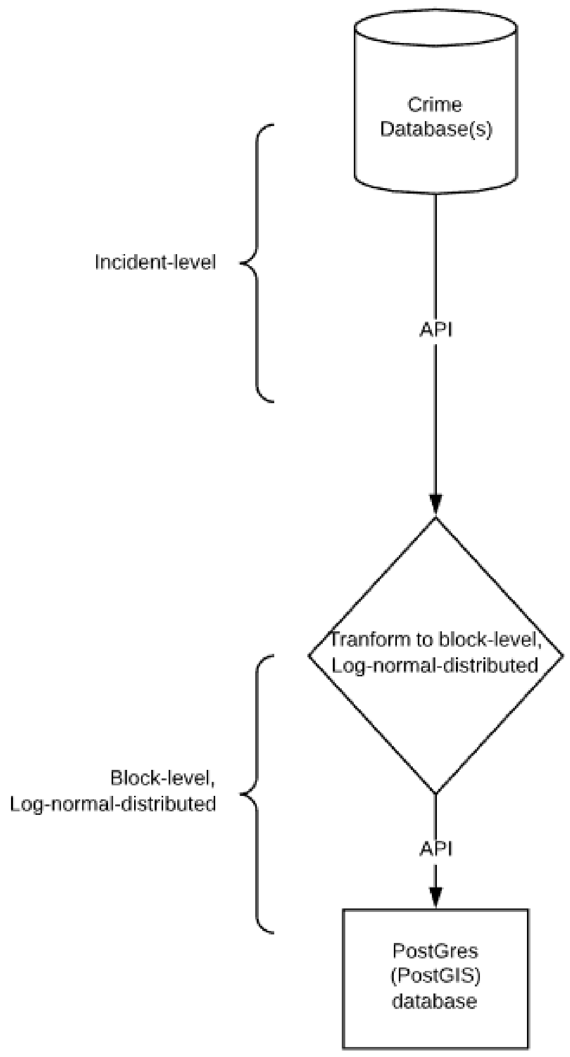
1 Overview

1.1 Reasons for development

This model can be explained in terms of requirements and solutions. The mobile application called **Ask Scout** ("Scout") developed by Civitas Secure ("CS") requires regular indicators of relative risk over a geospatial area, typically a city or municipality. Since this requirement is not currently satisfied by any data sources or products on the market or known to the author, a data model known as **Arkadin** was developed as a solution.

1.2 High-level process

Incident-level crime data is gathered from various sources such as socrata.com, pulled into a local R computing environment, transformed into a block-level, log-normal distributed data structure, and finally posted to a PostGres/PostGIS database.


Crime Database(s)
Incident-level
API
Tranform to block-level, Log-normal-distributed
Block-level, Log-normal-distributed
API
PostGres (PostGIS) database

1.3 Technology

While this model could be built with a number of computing languages, the R language was the primary programming language utilized. Various R packages such as SF, SP and Tigris provide the GIS support needed within the model.

2 Process

In this section, we will document the entire model from start to finish, ingestion to data output. The model can be separated into three distinct operations: *Munge*, *Arkadin*, and *Post*.

2.1 Munge operation

The munge operation ingests the incident-level data, normalizes the data structure, and performs classification with a machine learning model.

The incident-level data enters the model as a separate data set for each city or municipality. This is due to the fact that each city reports their data independently, and the context of knowing a risk level within a particular city is optimal.

2.1.1 Ingestion

First, data is ingested into the local R computing environment via a connection to an API provided by the incident-level crime vendor - typically Socrata. The amount of data ingested for each separate city varies; the objective is to ingest a statistically-significant and representative sample size.

In order to produce a robust sample size and also limit complexities in the code base, a simple logic is employed of using a sample size of 10k records, or $1/4 \times population$, whichever is larger. In testing, the model has produced almost identical results from this sampling as it did using the full dataset.

2.1.2 Data structure normalization

After the data set is ingested, traditional data munging functions are applied, primarily using the *tidyverse* suite of R packages. Depending on the native data structure, columns may be renamed or dropped, data types may be modified, etc. This process tends to vary from city to city, depending on proclivities in the source data.

2.1.3 Classification with machine learning

At this point, the incident-level data set is tidy: columns are in a consistent format and data types are in-order. Socrata and other vendors do provide some level of classification for each incident, such as “assault”, “battery,” etc. Ask Scout requires all incident-level data to be classified as one of four types: personal, property, auto or social.

A naive bayes classifier machine learning model is employed at this stage to classify each incident as one of the four crime types. The crime description from the incident-level data is utilized as the input to the model.

See below for a quick look at the conditional probability model in Bayesian probability terminology. The model utilized is based off of the *quanteda* R package.

$$posterior = \frac{prior \times likelihood}{evidence}$$

A large set of training data was labeled and used as input to build the model used here. The trained model is used to classify all new inputs, specifically the incident descriptions.

This model creates a new field in the data set that indicates which type of crime that incident is attributed to. At this point, each of these data sets are then placed in a staging directory in flat file format in order to be immediately used as input for the next operation.

For more information on this Bayesian approach to document classification see:(Manning 2008).

2.2 Arkadin operation

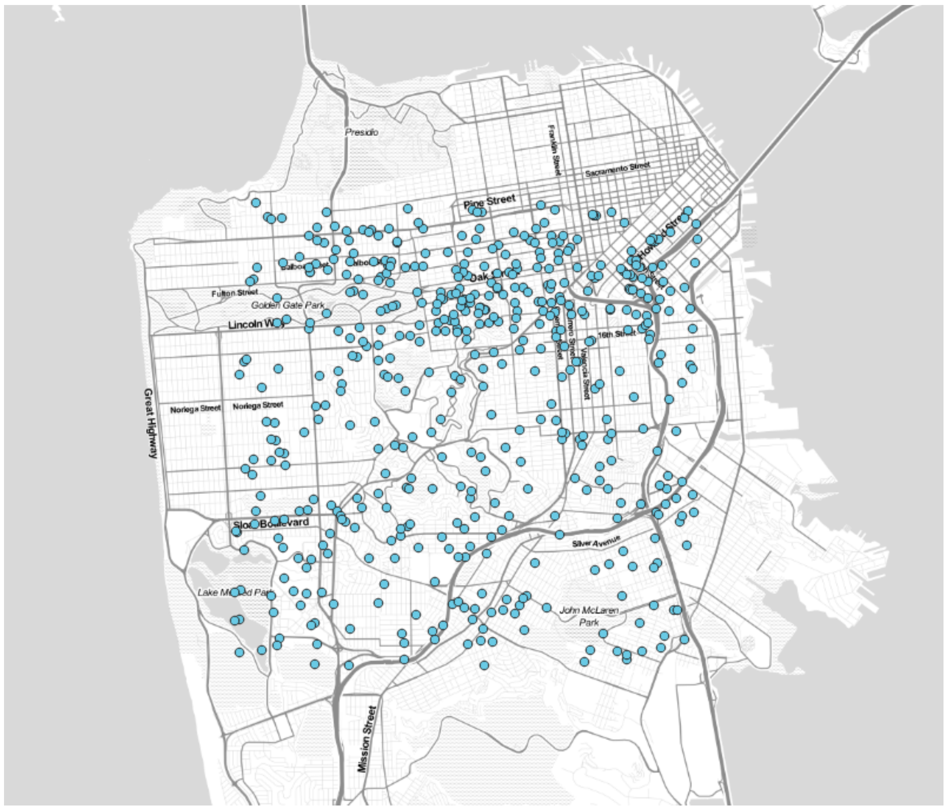
At this step, we have a collection of data sets with identical structures, which is essential for our next steps. In the Arkadin operation the model takes a data set and applies various transformations to the data so that Ask Scout can query this data from a database.

We will separate this Arkadin operation into five parts: *grid construction, split by type, point aggregation, log transformation, rescaling,* and *calculate centroids.*

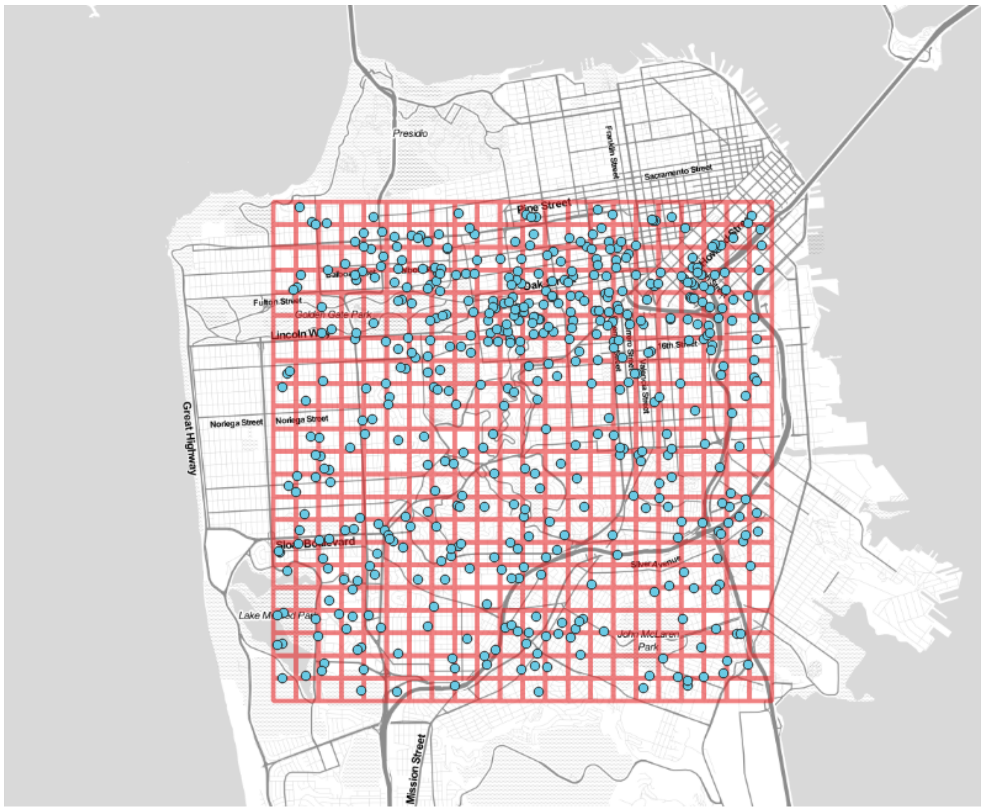
Note that for the rest of this operation we will use a single city as the context. This operation will be completed independently for each city or municipality.

2.2.1 Grid construction

For our example data, we will use San Francisco as our city of record. Our incident-level data could look something like this at this stage:



Each point represents one crime incident. We need to be able to aggregate these points to build a risk score for individual areas of the city. At this point, the model contructs a grid encompassing the entire area with centroids roughly 500 meters equidistant:



This grid or division of a windows into non-overalapping regions is called a *tesselation* (latin *tesserae*) and the extent of the grid is the *bounding region*, or *bounding box*.

2.2.2 Split by type

Next, the data set for San Francisco is split four-ways: by crime type (personal, property, auto, social). The data is split so that as it flows through the model, an independent Scout Score can be calculated for each crime type.

2.2.3 Point aggregation

For each crime type subset of the San Francisco data, we need to aggregate the points for each square in the grid, known as a *tile*. For each tile, we need to count the number of points that intersect it, using an intersection function. The operation can be summarized:

$$f(\tau) = \sum_{i=1}^{n} [p_i \cap \tau]$$

where τ is a singular tile and p is all points in the city.

These aggregations provide a count of crime incidents for each tile. These counts need further transformation.

2.2.4 Log transformation

In most cities, there is extreme clustering of crime incidents in certain areas. The distribution of these points very much follows a power distribution like the following:

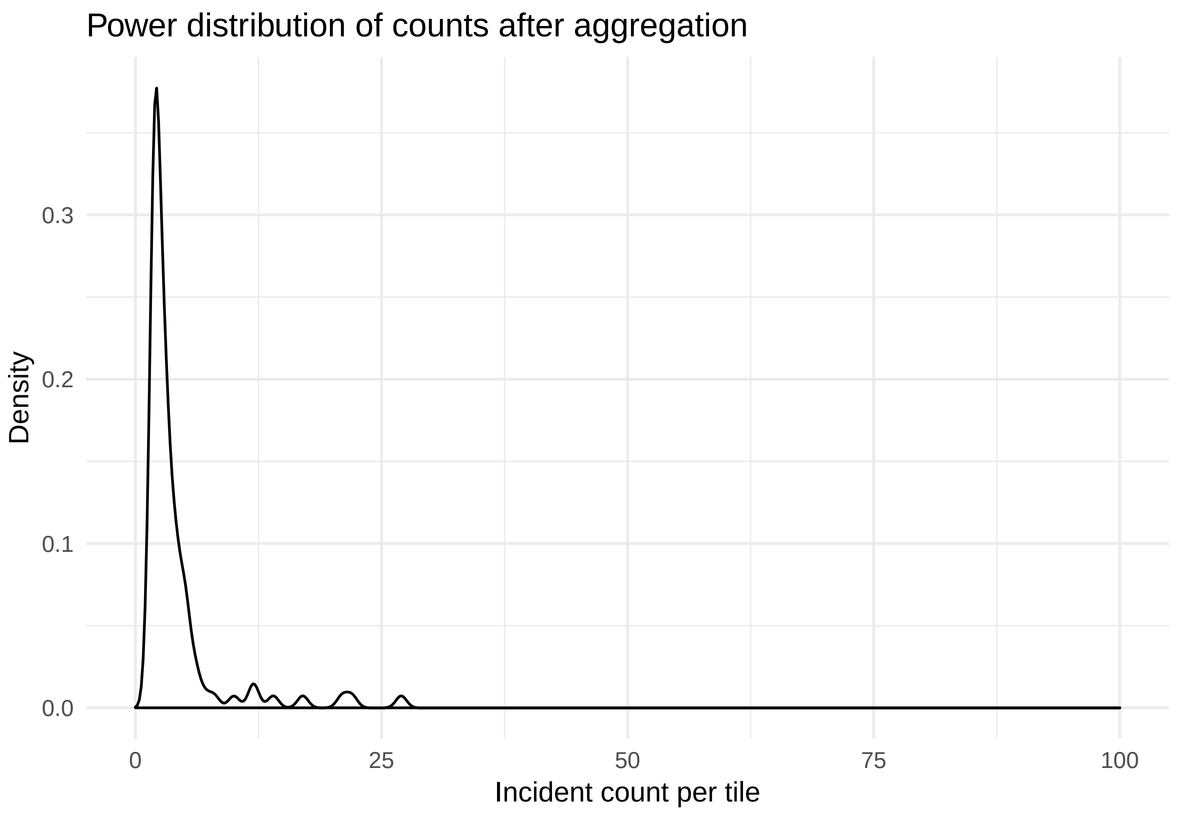


Table 1: Matrix of counts after aggregation

2	6	3	2	2	2	3	2	2	5
3	3	5	2	5	2	2	2	4	3
4	3	2	2	2	3	3	12	3	3
2	2	3	2	5	2	14	2	27	10
4	4	2	2	2	4	2	8	2	4
5	4	2	2	7	6	12	3	4	2
2	2	2	4	17	2	3	3	3	6
3	2	2	2	4	2	3	2	3	2
2	2	2	5	2	2	2	5	5	2
2	22	3	21	3	2	3	2	4	2

In this power-type distribution, extreme, infrequent outliers exist that tend to throw off any summary statistics for the tiles as a whole. This necessitates a transformation in the values to reflect a more moderate distribution - to put it plainly - shave off the magnitude of some of the large outliers to bring the distribution into a more normal configuration.

A *log* transformation - in our case, log10 - is now applied to the previously aggregated counts of incidents by tile. A log10 transformation is the inverse function to exponentiation, and is also known as the *common logarithm*. It is often used to transform data sets such as this one. Log10 can commonly be expressed as such:

$$log_b x = \frac{log_{10} x}{log_{10} b}$$

This log10 transformation transforms the matrix of values above into the following:

Table 2: Matrix of counts after log10

0.301	0.778	0.477	0.301	0.301	0.301	0.477	0.301	0.301	0.699
0.477	0.477	0.699	0.301	0.699	0.301	0.301	0.301	0.602	0.477
0.602	0.477	0.301	0.301	0.301	0.477	0.477	1.079	0.477	0.477
0.301	0.301	0.477	0.301	0.699	0.301	1.146	0.301	1.431	1.000
0.602	0.602	0.301	0.301	0.301	0.602	0.301	0.903	0.301	0.602
0.699	0.602	0.301	0.301	0.845	0.778	1.079	0.477	0.602	0.301
0.301	0.301	0.301	0.602	1.230	0.301	0.477	0.477	0.477	0.778
0.477	0.301	0.301	0.301	0.602	0.301	0.477	0.301	0.477	0.301
0.301	0.301	0.301	0.699	0.301	0.301	0.301	0.699	0.699	0.301
0.301	1.342	0.477	1.322	0.477	0.301	0.477	0.301	0.602	0.301

These compressed values are much easier to work with.

2.2.5 Rescaling

After the log10 transformation, we have a data set with moderated values, that is, compressed on the long tail. The last transformation is necessary to develop a "Scout Score" of 0 to 100. This Scout Score is the proprietary name for the final output from this model.

To transform this now-log10-normalized data, we need to use a re-scaling function.

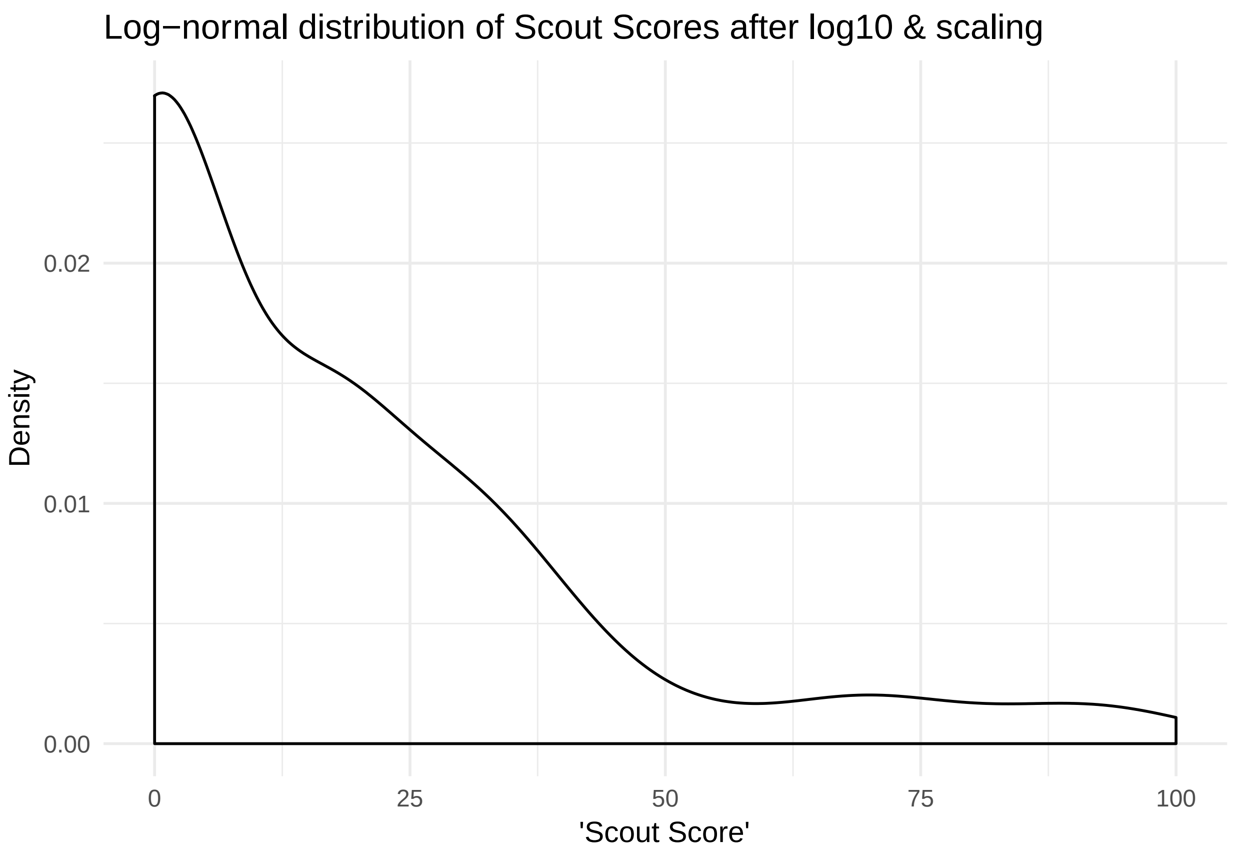
$$f(x) = \frac{min(x)}{max(x) - min(x)} \times 100$$

where x is a vector of log10-normalized data. This provides a 0-100 score for each tile in the data set.

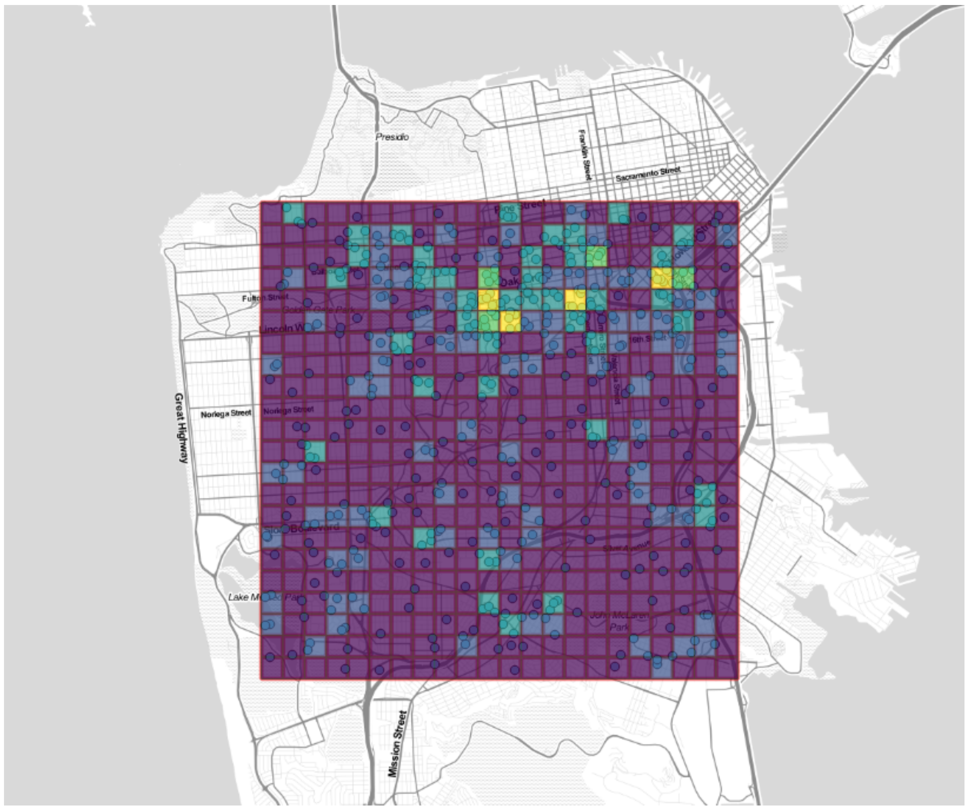
Table 3: Matrix of counts after scaling

0	42	16	0	0	0	16	0	0	35
16	16	35	0	35	0	0	0	27	16
27	16	0	0	0	16	16	69	16	16
0	0	16	0	35	0	75	0	100	62
27	27	0	0	0	27	0	53	0	27
35	27	0	0	48	42	69	16	27	0
0	0	0	27	82	0	16	16	16	42
16	0	0	0	27	0	16	0	16	0
0	0	0	35	0	0	0	35	35	0
0	92	16	90	16	0	16	0	27	0

At this stage, we have a much better log-normal distribution of values:



If you were to make a heatmap using these values, you would see a well-distributed display of risk like the following:



where the warmer colors indicate higher values, and the cooler colors indicate lower values.

2.2.6 Calculate centroids

At this stage, for each of the tiles in the bounding box, there will be a value between 0-100. This is the indicator of risk for that area, determined by apriori association. The final step in this operation is to calculate the centroids of each tile. This allows us to store POINT data instead of the more memory-intensive POLYGON data that it is currently in.

A centroid is the point in the tile where $(\overline{x}, \overline{y})$, that is, the mean of all points x and y within that tile. This can be defined more formally as such:

$$C = (\frac{b}{2}, \frac{h}{2})$$

where b is base of tile and h is height of tile.

2.2.7 Context

So far we've calculated the *overall* measure of risk for each tile, with a timeframe of nine months, on average. But, the application requires aggregation by day of week and by hour. Therefore, we must perform the above operation over another grouping of the same base data.

2.2.7.1 Daily aggregation

The goal of the daily aggregation is to provide a measure of risk for each day of the week. To accomplish this, the model also aggregates the base data by day of week. The output is a risk score for each tile (centroid), by each day of the week.

2.2.7.2 Hourly Aggregation

The same applies for hour of the day; the base data is aggregated by the hour of the day and output is the same.

2.3 Post operation

The terminal operation of the model involves moving the data to a database where the application can query it. The script uploads the output of the previous operations to a PostGres/PostGIS database through a custom API.

References

Manning, Schutze, Raghavan. 2008. "An Introduction to Information Retrieval." Journal Article. *Cambridge University Press*, January, 1–25.