Gravity Model

Twitter Data from NYC

Overview

1. Probabilistic Weight

Construct a network with the probabilistic weight of link(a,b) defined as

$$link\ weight(a,b) = \Sigma_c \frac{t(c,a) \cdot (t(c,b) - \delta(a,b))}{T \cdot (t(c) - 1)}$$

where

- *t(c,a)* denotes the total number of tweets that user *c* has posted at location (in our case, zipcode) *a*
- $t(c) = \Sigma_a t(c,a)$
- $T = \Sigma_c t(c)$

2. Gravity Model

On the other hand, the weight of links can be modeled as

$$link\ weight(a,b) = k \cdot w^{out}(a) \cdot w^{in}(b) \cdot f(d(a,b)),$$

where

- $w(x)$ is the weight or customized centrality of the node x
- $f(d(a,b))$ denotes function with respect to distance between location a and b, usually it dacays as distance increases.

Formulae

So after calculating all $link(a, b)$, we run linear regression by taking logarithm on both sides

A few notes:

- **Original model** uses link weight directly from the formula:

$$log(weight_{a,b}) \sim log(w_a^{out}) + log(w_b^{in}) + log(f(d(a, b))$$

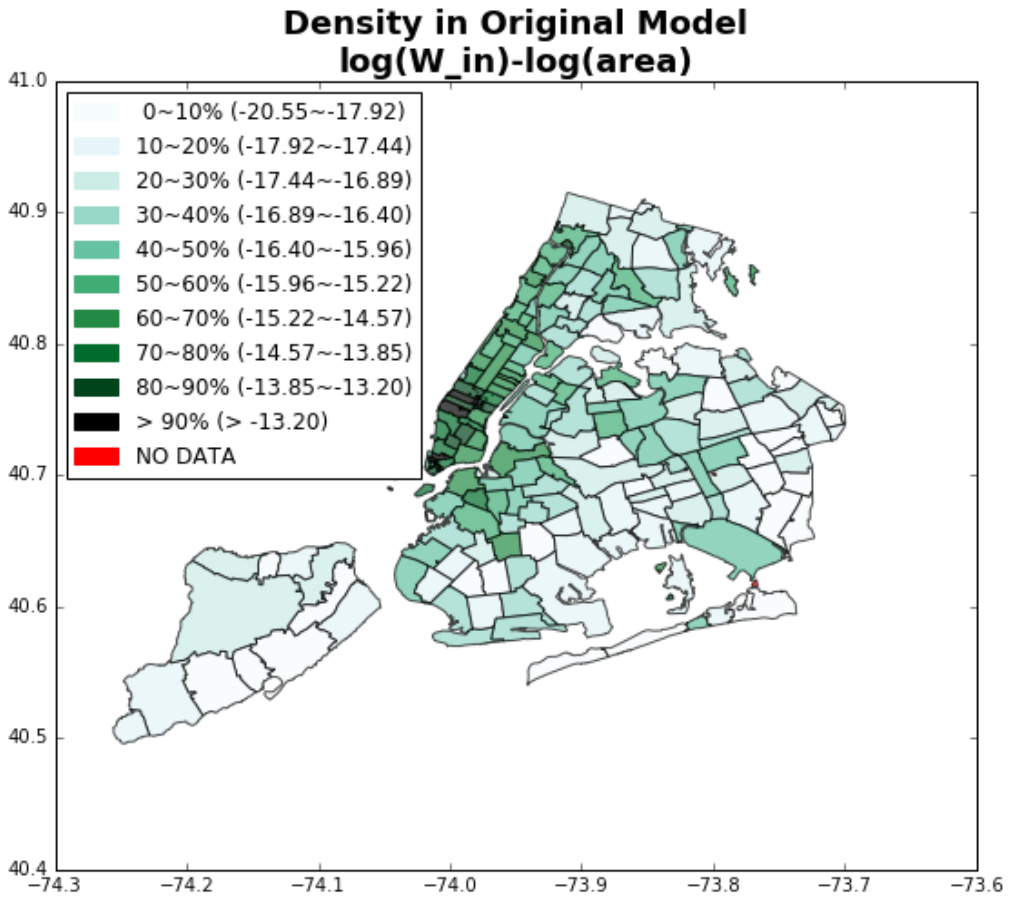
- **Adjusted model** uses (link weight) * (number of tweets from origin) instead:

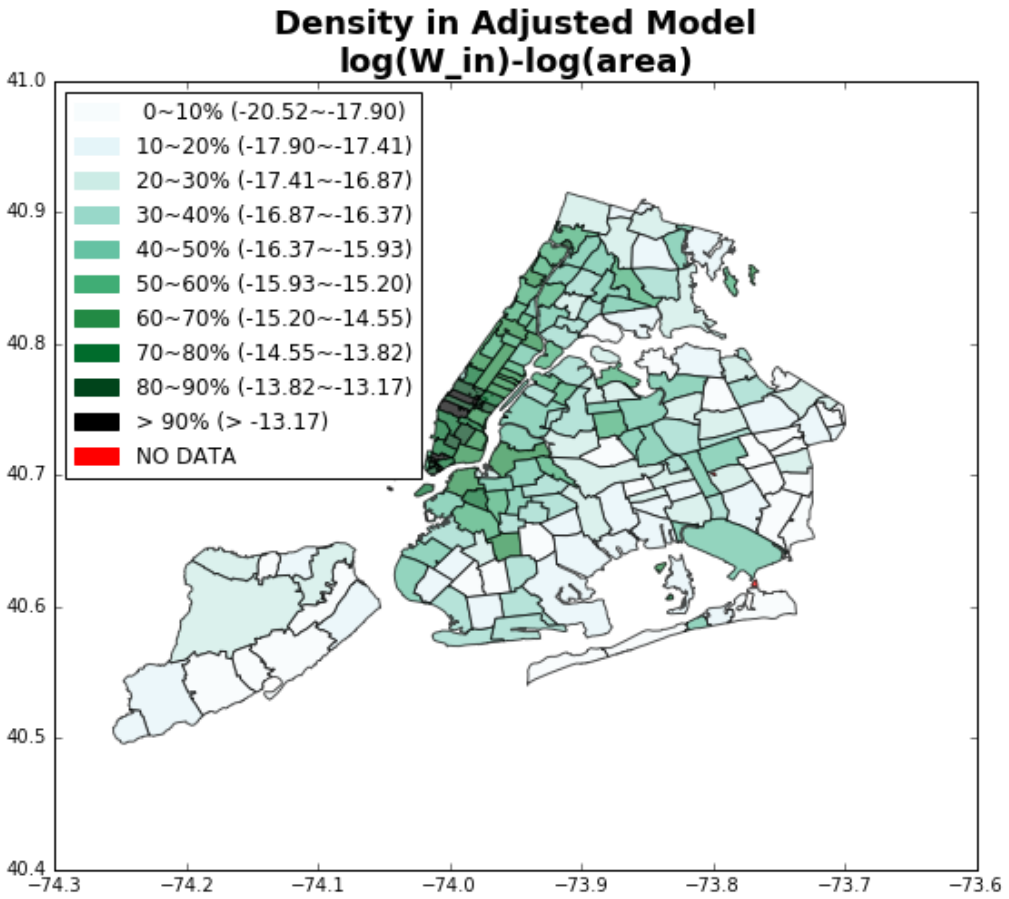
$$log(weight_{a,b}) + log(number\ of\ tweets\ at\ a) \sim log(w_a^{out}) + log(w_b^{in}) + log(f(d(a, b))$$

- NYC has about 240~250 zip codes, so we have 242 terms each(from twitter data) to be fit, for both W^{out} and W^{in}
- For $f(d(a, b))$, we used a custom binned function to seperate all distance data into 100 subcategories based on its position between percentiles, so we have to fit an additional amount of 100 of distance bin, then observe the relationship between the coefficients and the distance within each bin

On the following pages, we use heatmap to showcase both models for comparison. As we can see from the plot, adding $log(number\ of\ tweets\ at\ a)$ does significantly change the look of W_a^{out}, but W_b^{in} appears little difference

Divide by the area (W_b^{in})





Divide by the area (W_a^{out}) – for comparison