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:

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

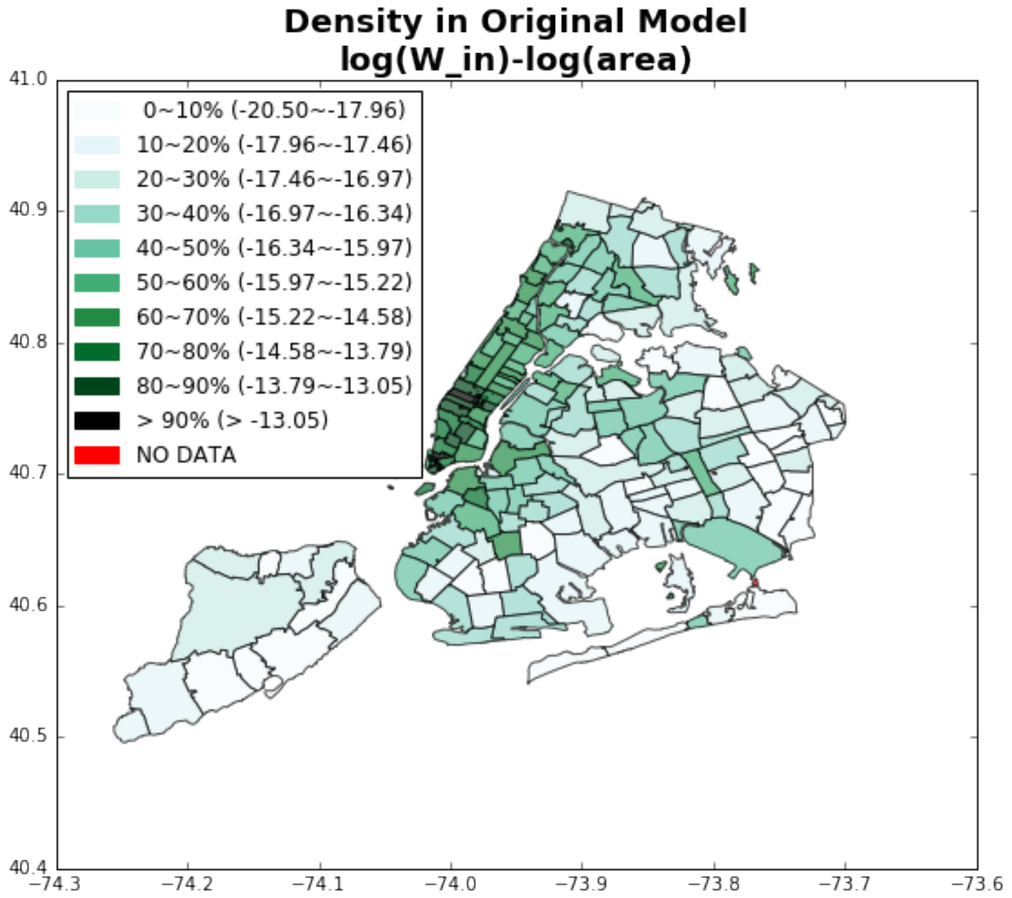
- NYC has about 240~250 zip codes, so we have 242 terms each(from twitter data) to be fit, for both W^{out} and W^{in}
- For $f(d(a, b))$, we used a custom binned function to seperate all distance data into 100 subcategories based on its position between percentiles, so we have to fit an additional amount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

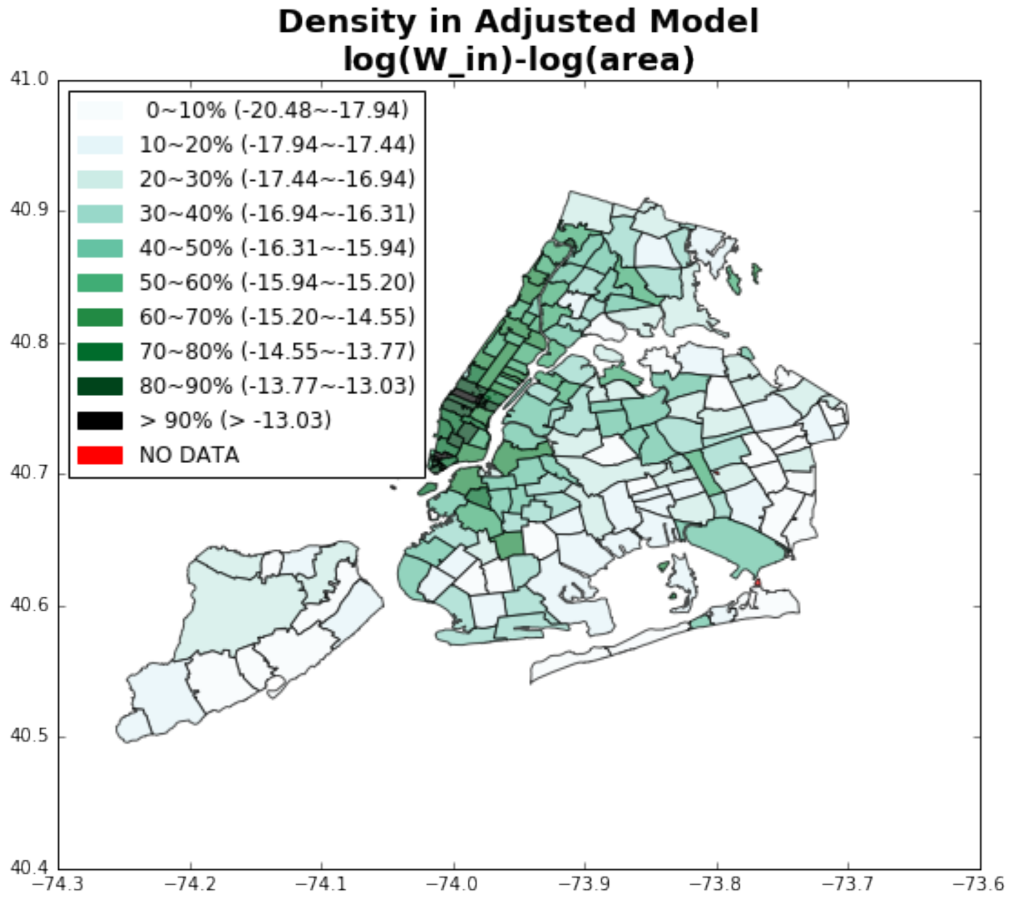
On the following pages, we use heatmap to showcase both models for comparison. As we can see from the plot, adding $log(number\ of\ tweets\ at\ a)$ does significantly change the look of W_a^{out}, but W_b^{in} appears little difference

W_b^{in}:

Divide by the area

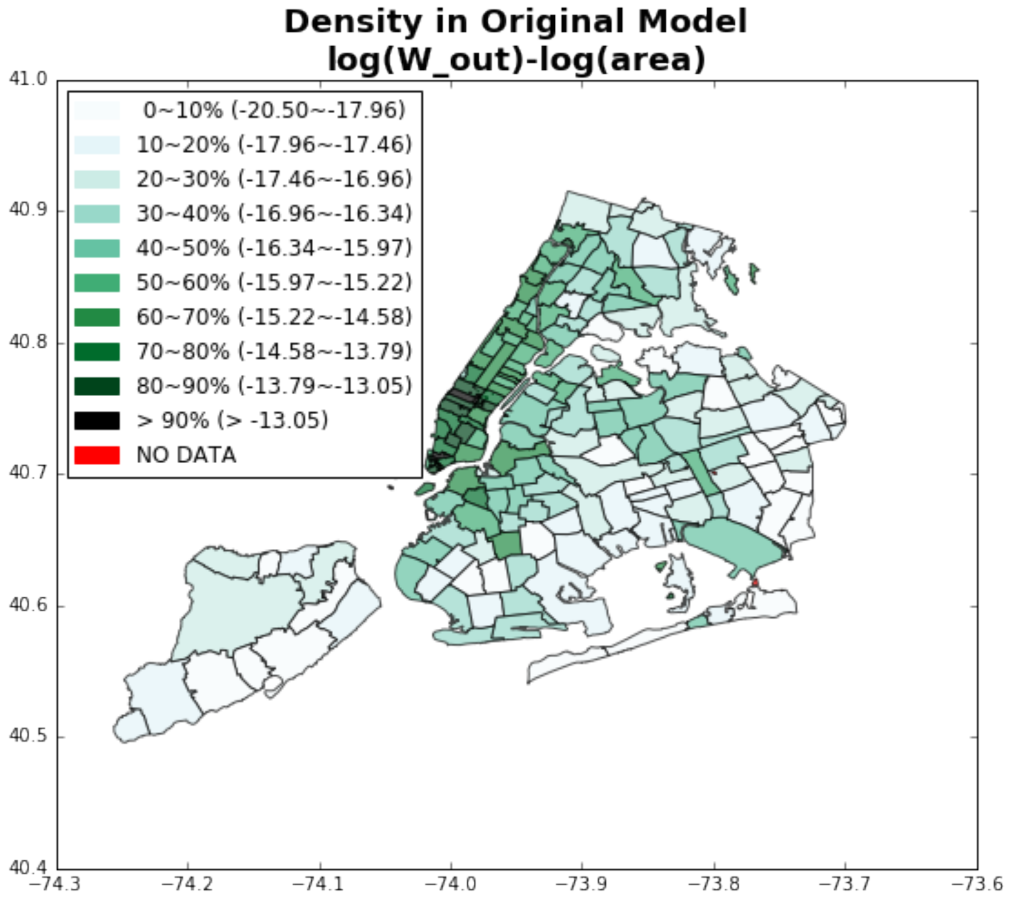
Not showing much difference if we add the log of number of tweets

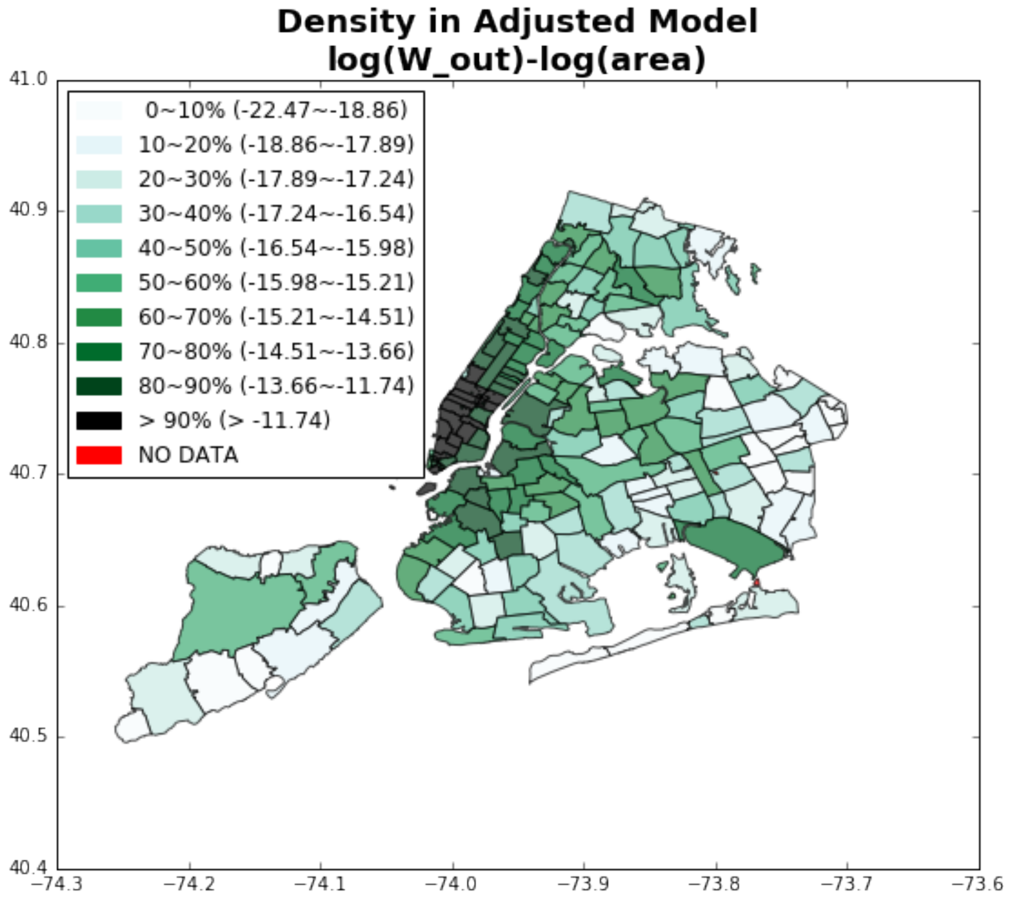




W_a^{out}:

Divide by the area as well
The absolute quantities don’t change significantly but the distribution does, resulting in more darker coverage all over the city on different level



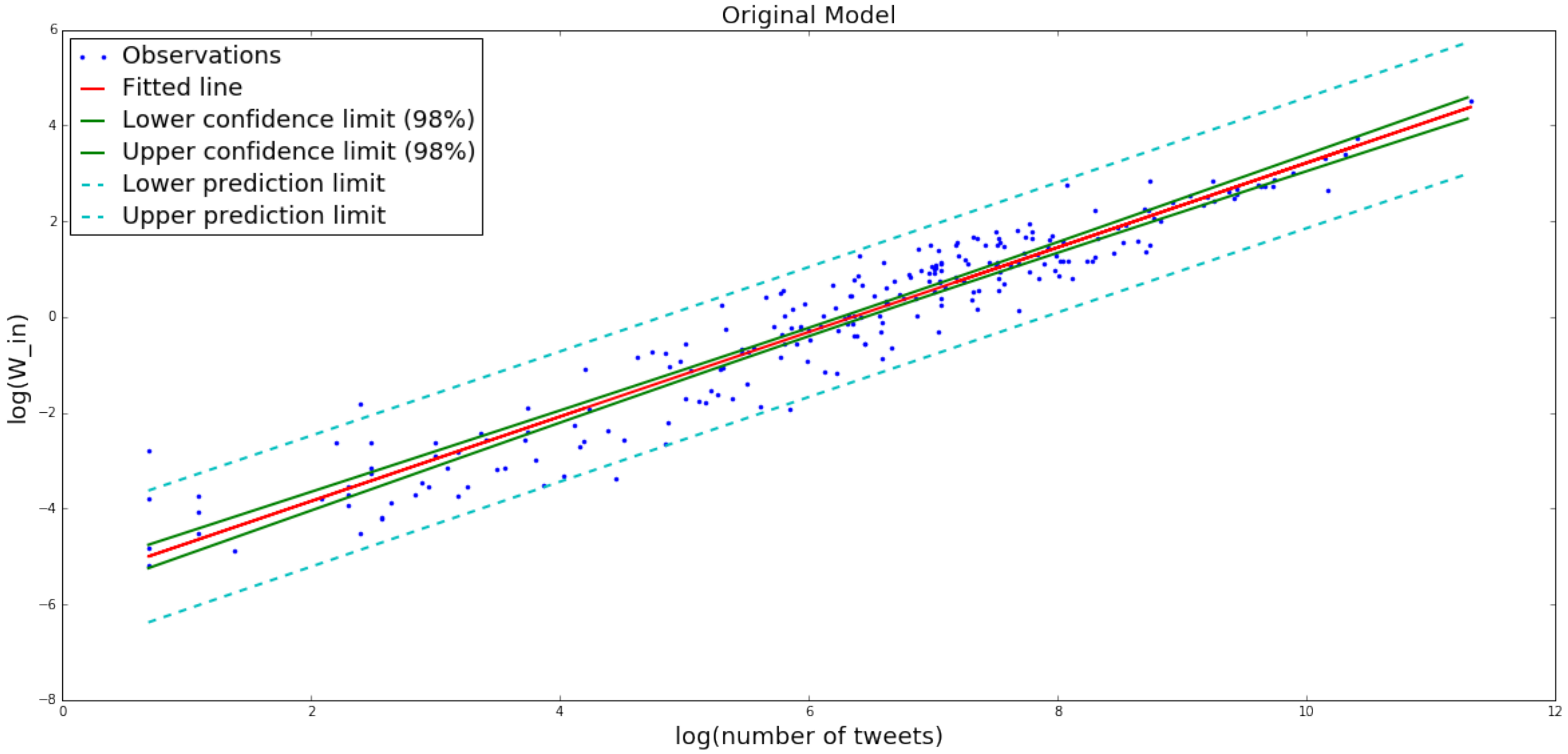


The relationship between W_b^{in} and #Tweets (Original Model)

$\log(W_b^{in})$ = 0.883218 • log(#Tweets) -5.614741 or W_b^{in} = 0.003643 •$(\#Tweets)^{0.883218}$

The R square of this model is 0.913803

The model is under linear as the exponent is approximately 0.883218, with 95% CI upper limit 0.9175

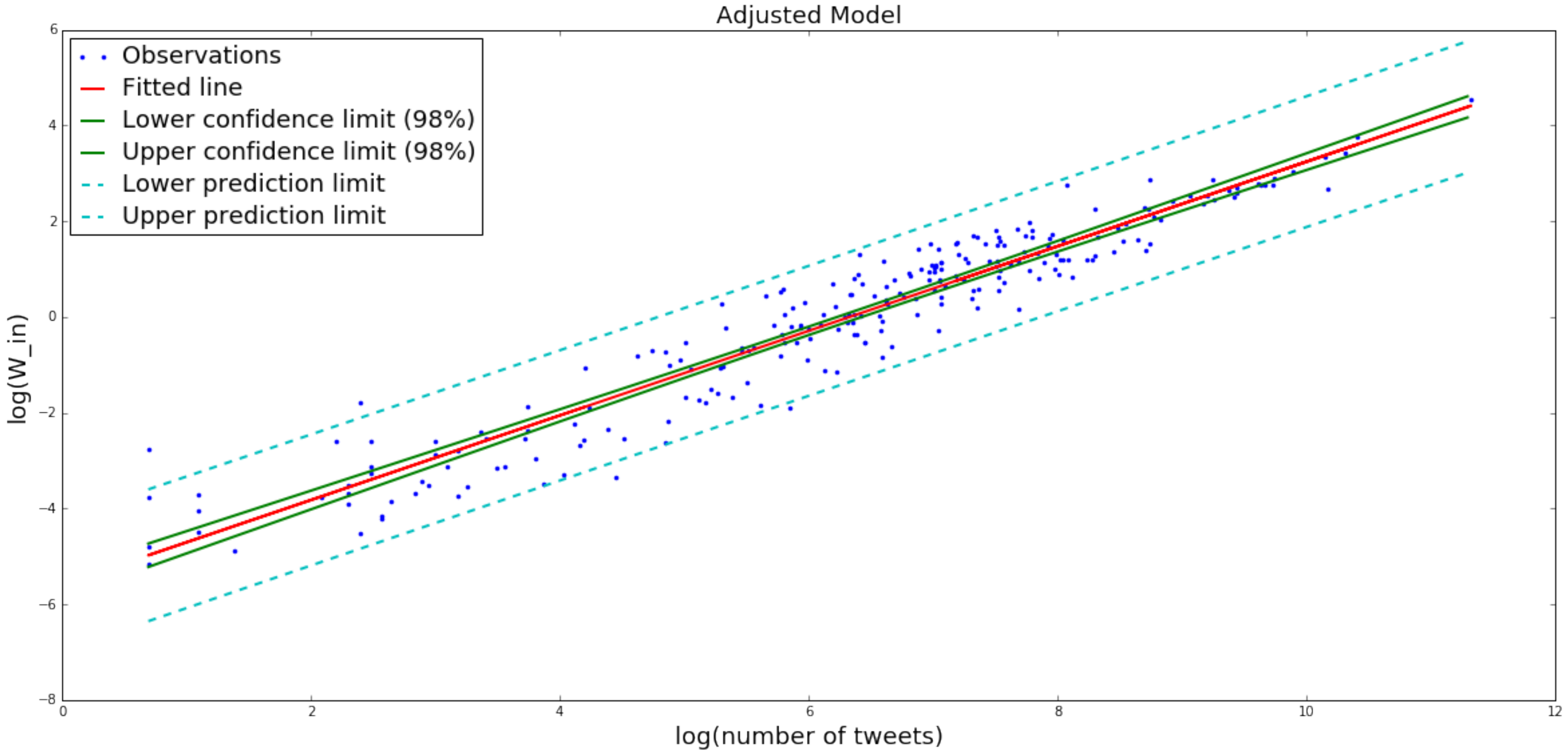


The relationship between W_b^{in} and #Tweets (Adjusted Model)

$\log(W_b^{in})$ = 0.883218 • log(#Tweets) -5.614741 or W_b^{in} = 0.0037358 • $(\#Tweets)^{0.883218}$

The R square of this model is 0.913803

The model is under linear as the exponent is approximately 0.883218, with 95% CI upper limit 0.9175



Further exploration

We've focused on the dataset before Dec. 19,2015 to avoid massive empty data frames between Dec.19, 2015 and Feb. 1, 2016

- 1. Examine two distance function models

$$\frac{1}{f(x)} = ax + b \;\; and \log(f(x)) = a \cdot \log(x) + b$$

or equivalently

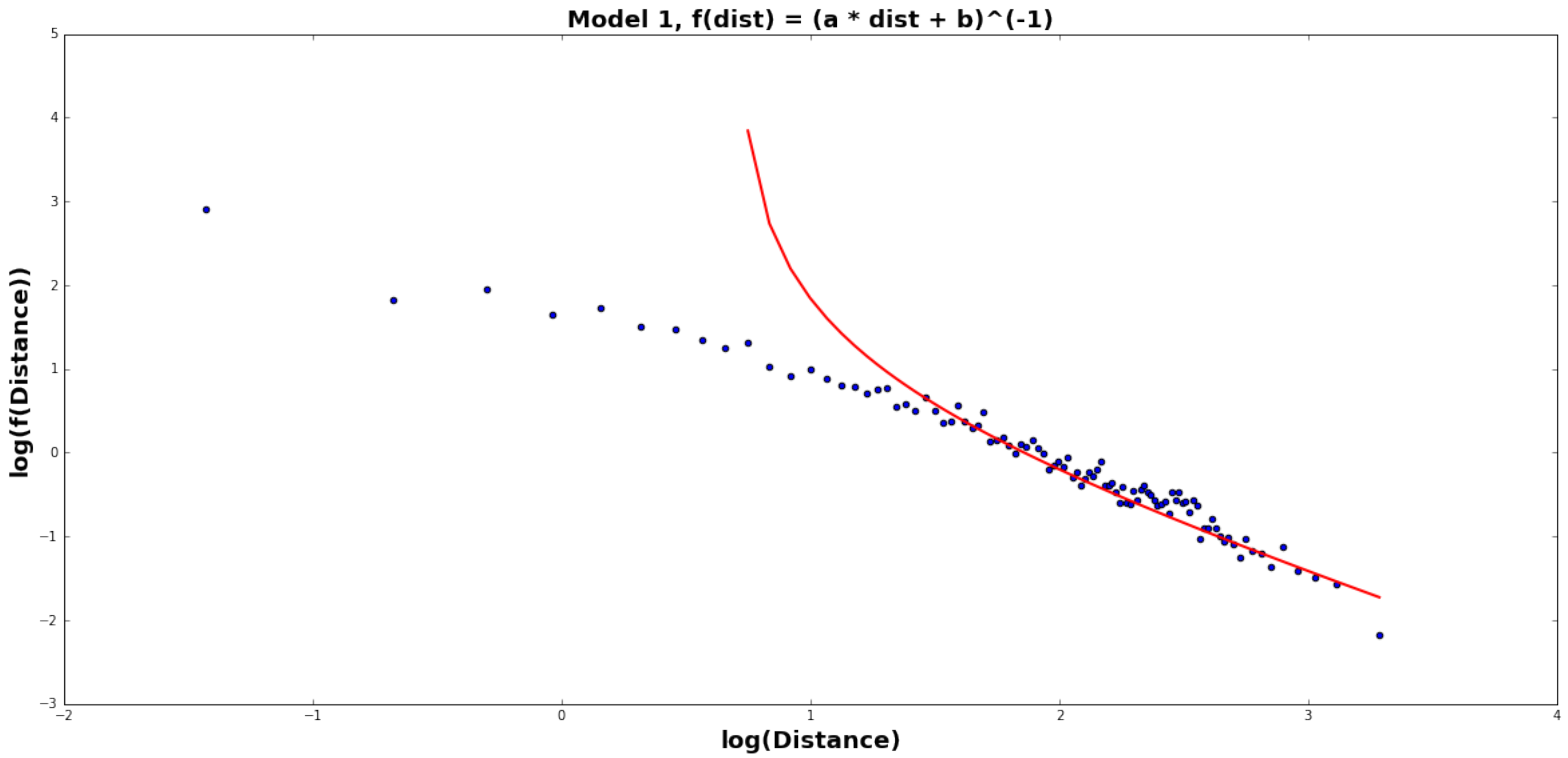
$$f(x) = \frac{1}{ax+b} \;\; and \;\; f(x) = e^b \cdot x^a$$

where

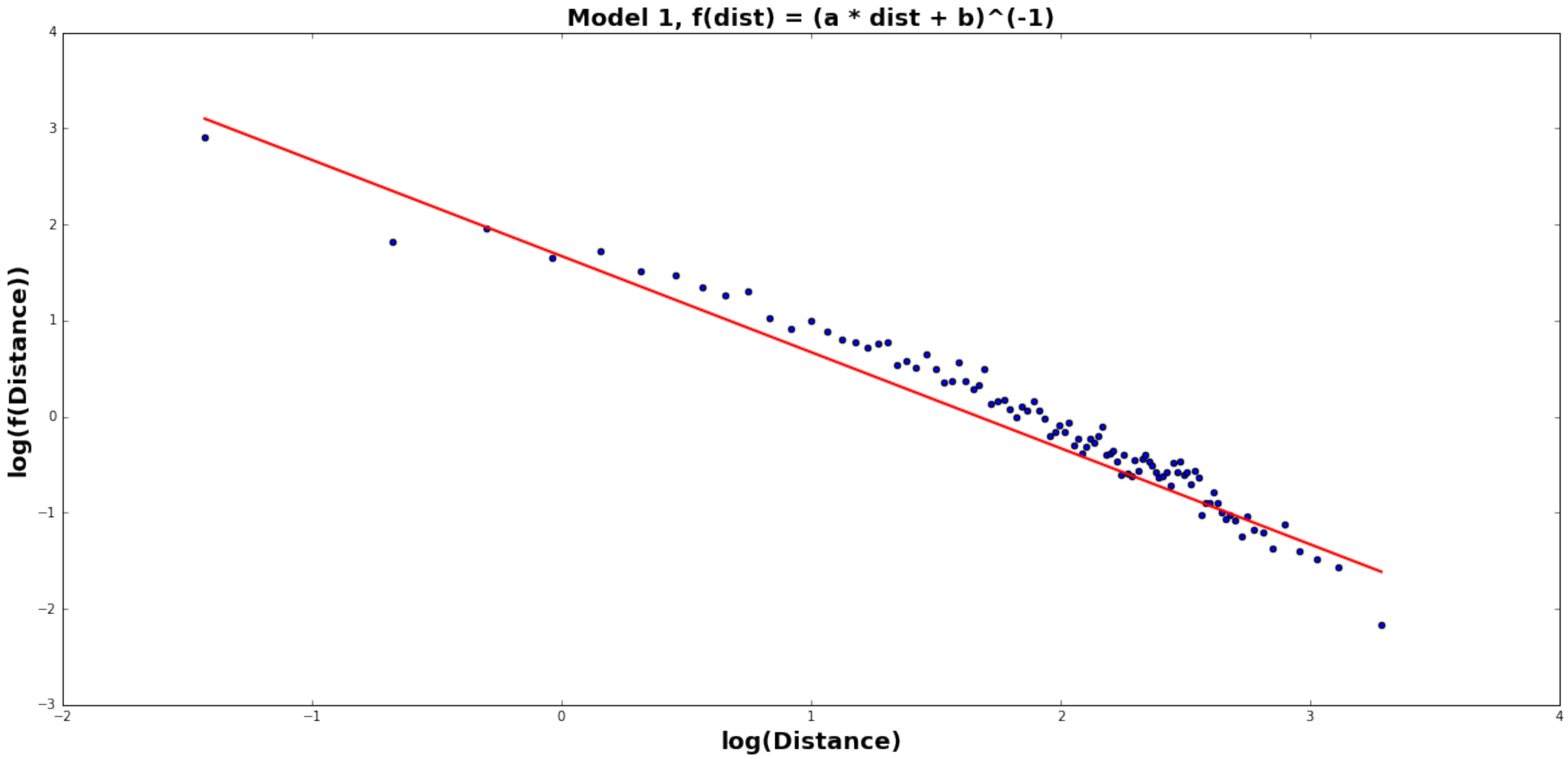
  - log(f(x)) is the fitted coefficients for each distance bin in our previous *adjusted model*
  - x denotes the distance (we've converted the average distance within each bin to this variable)

- 2. Experiment by dropping the constant term b
- 3. Check the exponent a for each day, see if they vary drastically, and compare it with the overall fitting number.

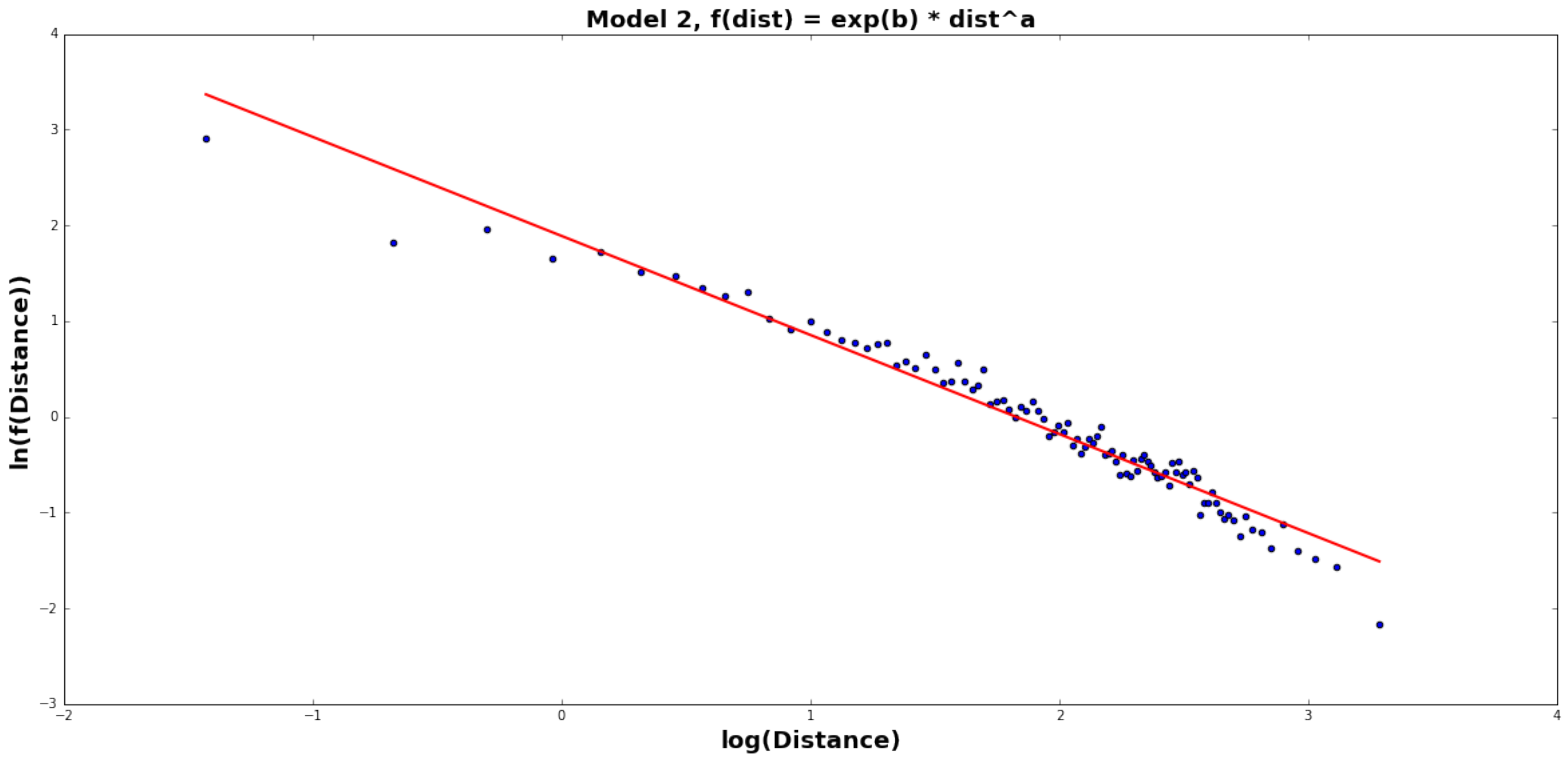
Model 1: $f(x) = \frac{1}{ax+b}$



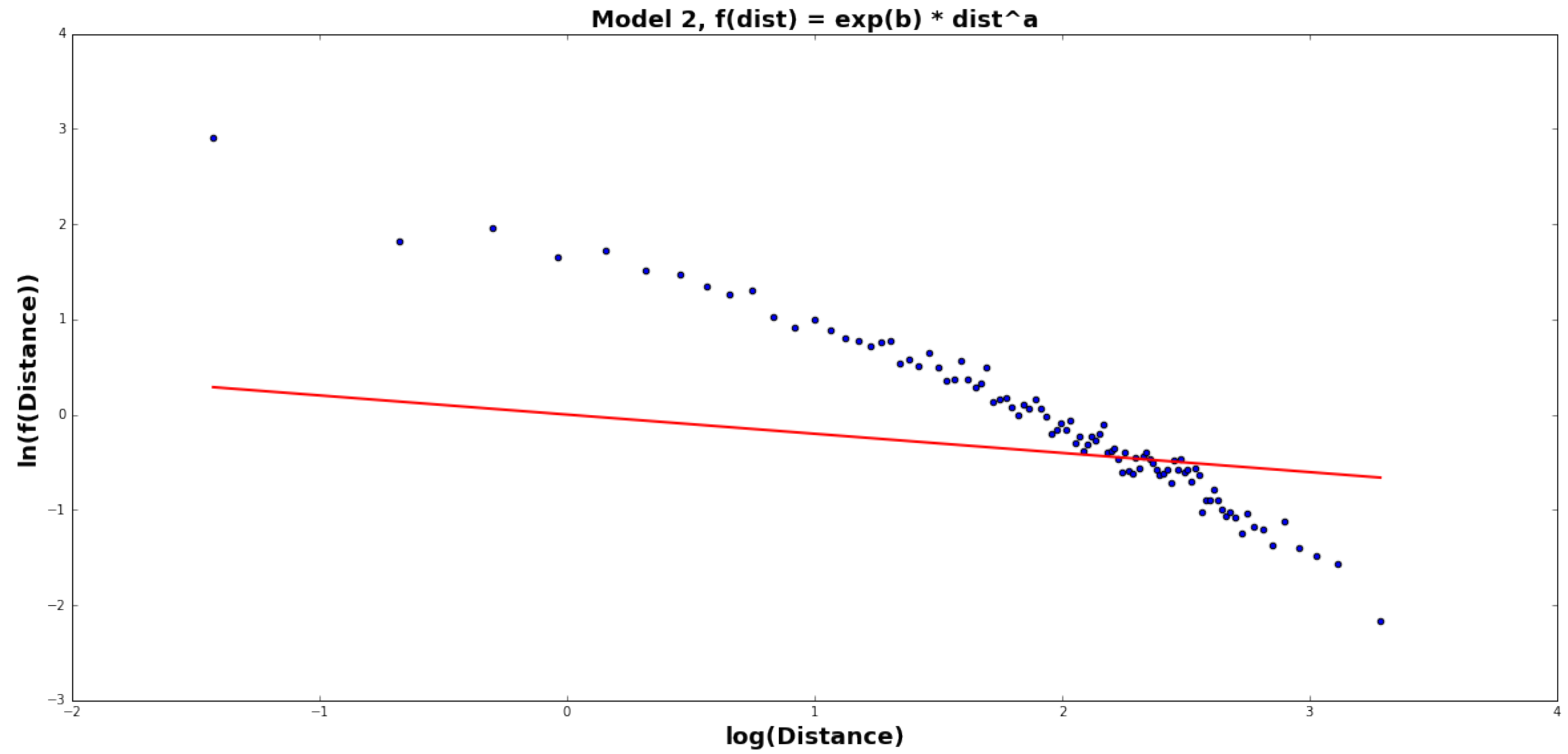
Model 1, drop b: $f(x) = \frac{1}{ax}$ (much better)



Model 2, $f(x) = e^{b} \cdot x^{a}$

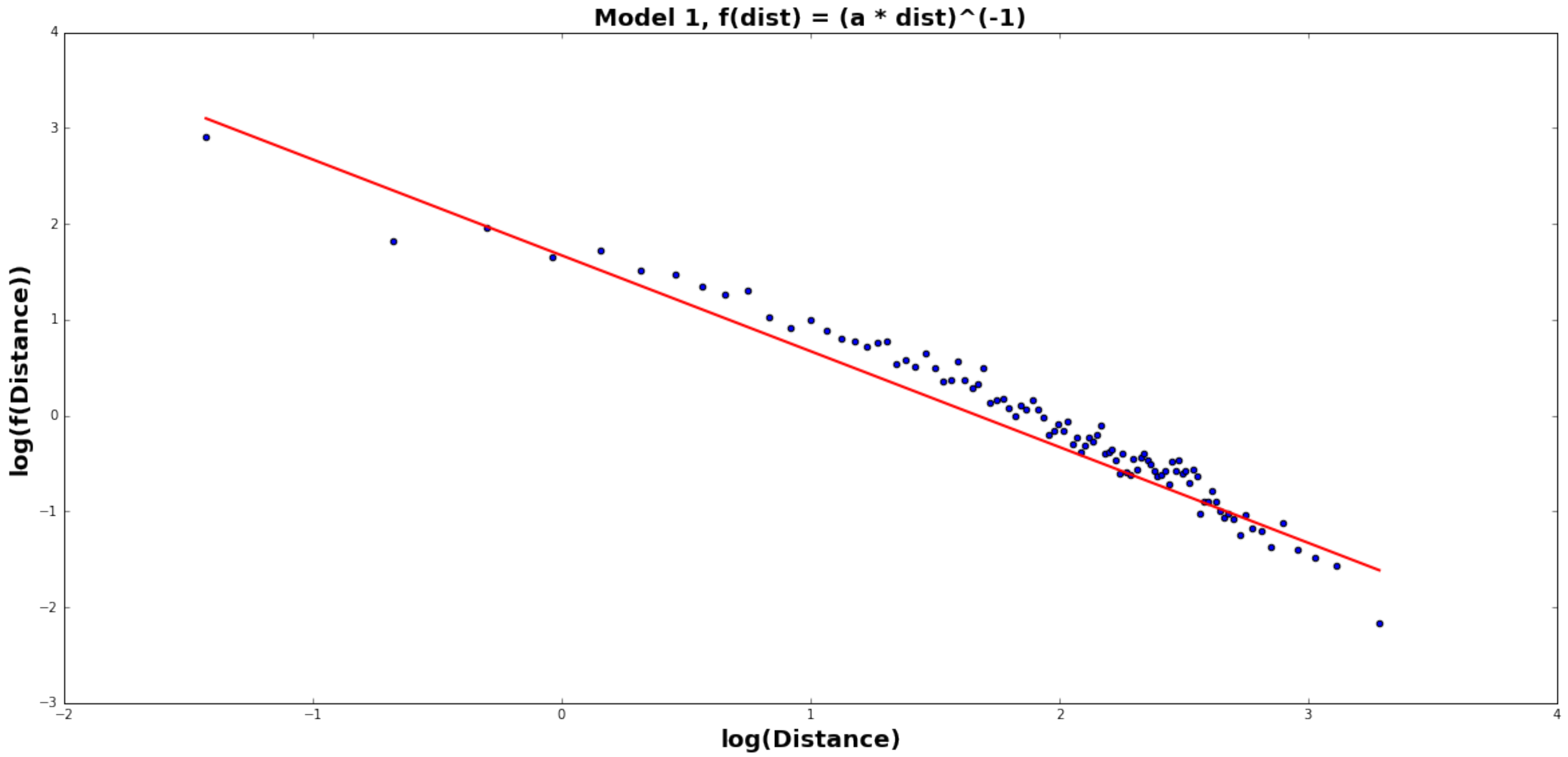


Model 2, drop b: $f(x) = x^a$ (gets worse)

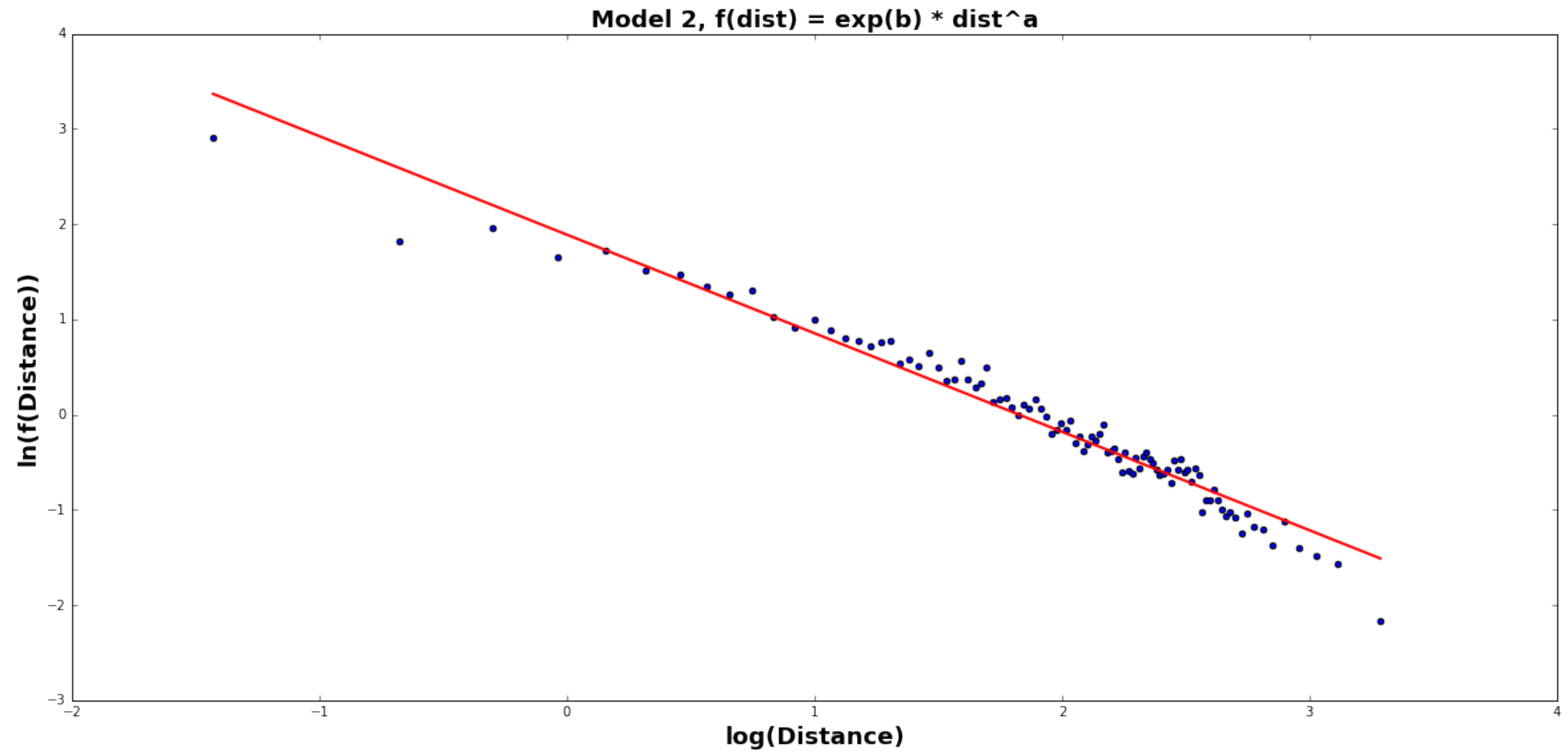


So we have two good options:

A. $f(x) = \frac{1}{ax}$ with a=0.188078, $R^2 = 0.940872$



B. $f(x) = e^{b} \cdot x^{a}$ with a=- 1.034442, b= 1.888632 , and $R^2 = 0.958063$

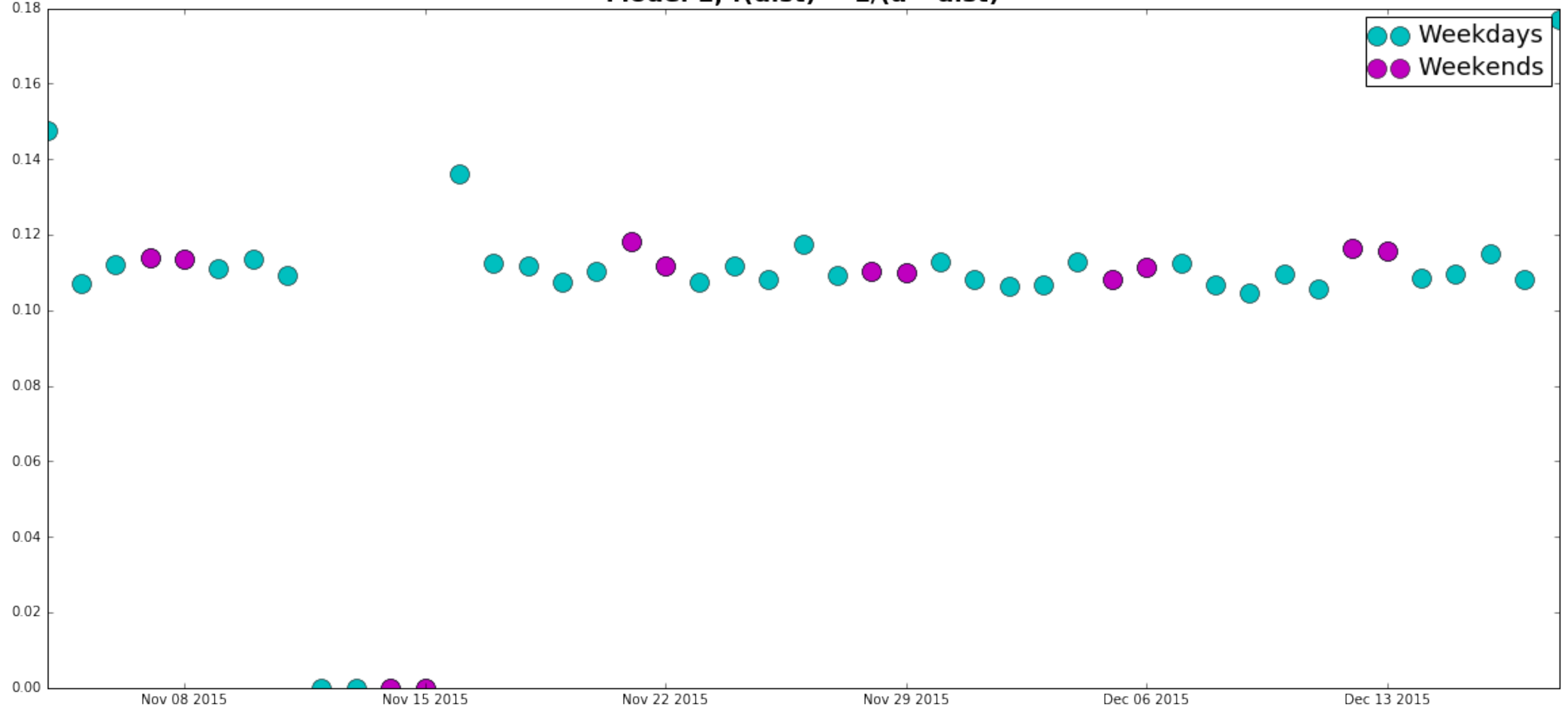


Time series ($f(x) = \frac{1}{ax}$)

For non-zero part	mean	std
weekdays_a	0.114109	0.014487
weekends_a	0.112952	0.003134

(Weekends perform much stable but with same average values)

Coefficient a
Model 1, f(dist) = 1/(a * dist)

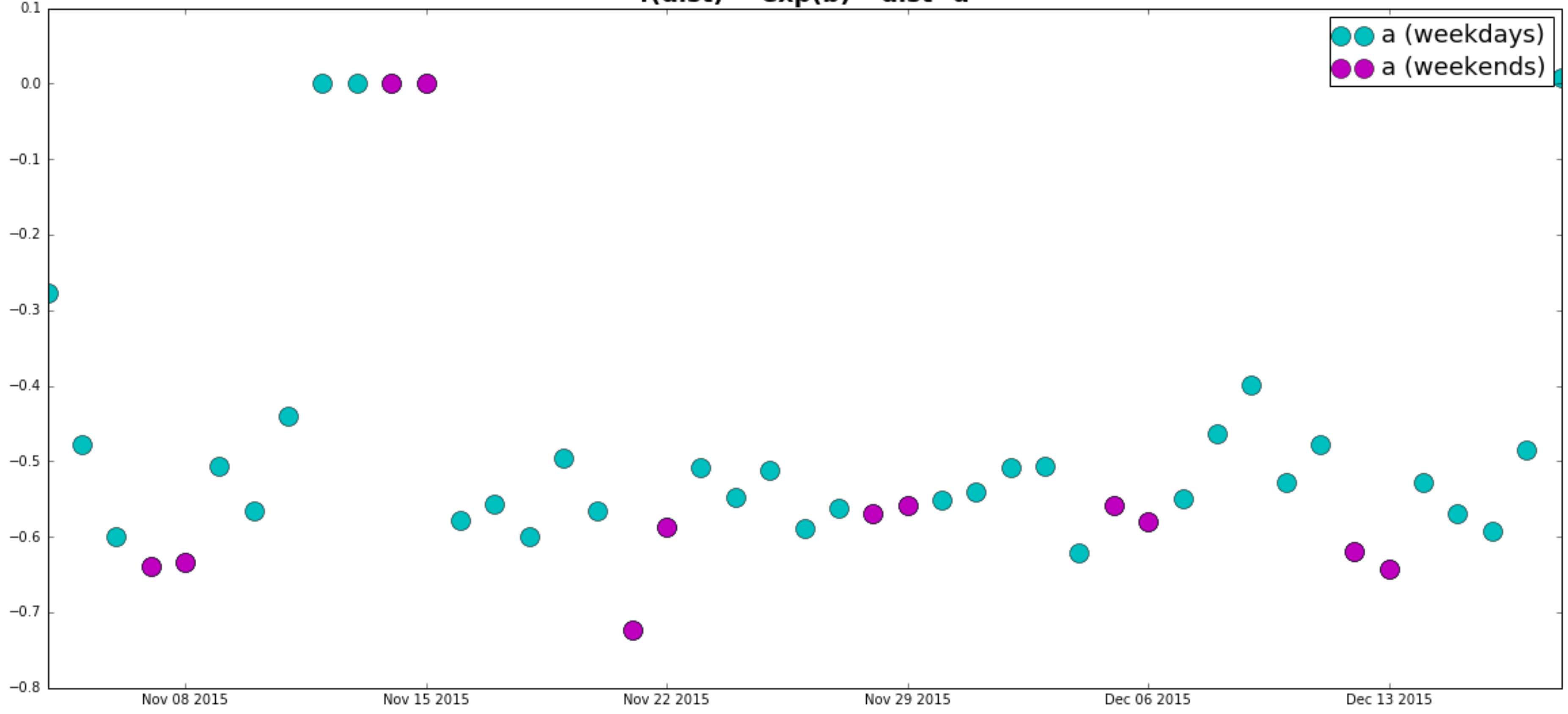


Time series ($f(x) = e^b \cdot x^a$)

For non-zero part	mean	std
weekdays_a	-0.50639	0.1170392
weekends_a	-0.61123	0.0515498

(Similar observation about a, but can't drop b without jeopardizing the accuracy of our model since b **changes everyday**)

The coefficients a and b (Black dots indicate weekend)
f(dist) = exp(b) * dist^a


Nov 08 2015
Nov 15 2015
Nov 22 2015
Nov 29 2015
Dec 06 2015
Dec 13 2015
a (weekdays)
a (weekends)