Kyu Cho
5/28/2016

# Title

Identify the correct place for check ins

# Background

The goal of this competition is to predict which place a person would like to check in to. For the purposes of this competition, Facebook created an artificial world consisting of more than 100,000 places located in a 10 km by 10 km square. For a given set of coordinates, your task is to return a ranked list of the most likely places. Data was fabricated to resemble location signals coming from mobile devices, giving you a flavor of what it takes to work with real data complicated by inaccurate and noisy values. Inconsistent and erroneous location data can disrupt experience for services like Facebook Check In.

# Data

- row_id: id of the check-in event
- x y: coordinates
- accuracy: location accuracy
- time: timestamp
- place_id: id of the business, this is the target you are predicting

# Research Question

Explanatory variables (row_id, x y, accuracy, time), Response variable (place_id)
Is there any relationship that explanatory variables effect on response variable?

# Methods

In this competition we're given around 30 million (simulated) check-ins on Facebook in a 10km by 10km grid. The goal is to build a model that predicts what business a user checks into based on spatial and temporal information. The tricky part here is that there are around 100k different classes (place_id's) so most supervised learning techniques won't work on the entire dataset. However most classes are clustered in only certain parts of the grid so the idea I'll pursue here is to select a small-ish square within the grid and try to see if we can do better within the small square. First I'll do some exploratory data analysis in the smaller square then I'll use a random forest algorithm for prediction and finally, I'll analyze the results. Statistical analysis we are going to use here is mainly through the visualization and some inferences.

Kyu Cho
5/28/2016

# Explanatory Analysis

```
## Observations: 29,118,021
## Variables: 6
## $ row_id   (int) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ x        (dbl) 0.7941, 5.9567, 8.3078, 7.3665, 4.0961, 3.8099, 6.333...
## $ y        (dbl) 9.0809, 4.7968, 7.0407, 2.5165, 1.1307, 1.9586, 4.372...
## $ accuracy (int) 54, 13, 74, 65, 31, 75, 13, 85, 3, 65, 6, 73, 64, 62,...
## $ time     (int) 470702, 186555, 322648, 704587, 472130, 178065, 66682...
## $ place_id (dbl) 8523065625, 1757726713, 1137537235, 6567393236, 74406...
```

A few notes:

- row_id seems to be … a row ID. It is TRUE that the number of unique row_ids is the same as the number of rows in the data frame.
- x is presumably bounded between [0, 10] as the x-axis on the 10-km square.
- y looks to be the same as x, just the other dimension.
- accuracy is intersting: it's all over the place. The smallest value is 1.00; the biggest value is 1,033.00. We'll have to look into that.
- time has no units. Since Facebook notes that time an accuracy are "intentionally left vague in their definitions.", we will have to look into that.
- place_id is probably a unique identifier. There 108390 unique values.

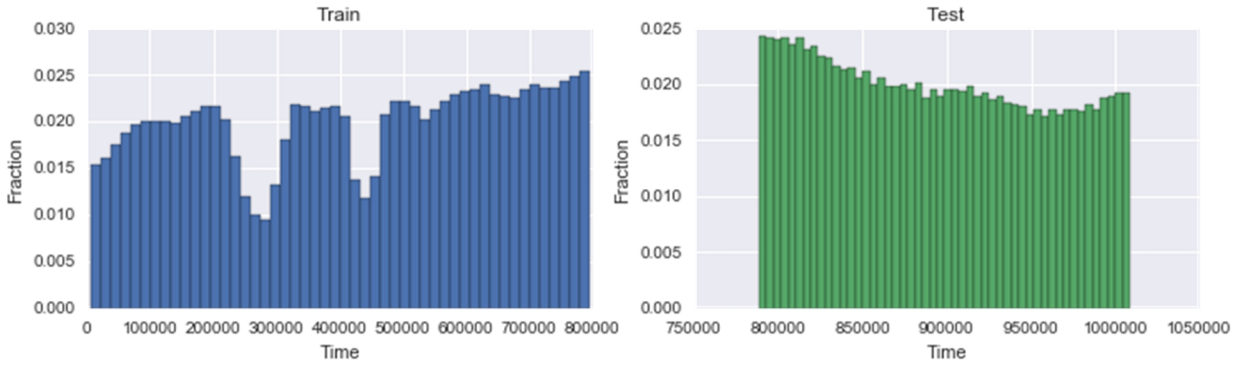
# Time

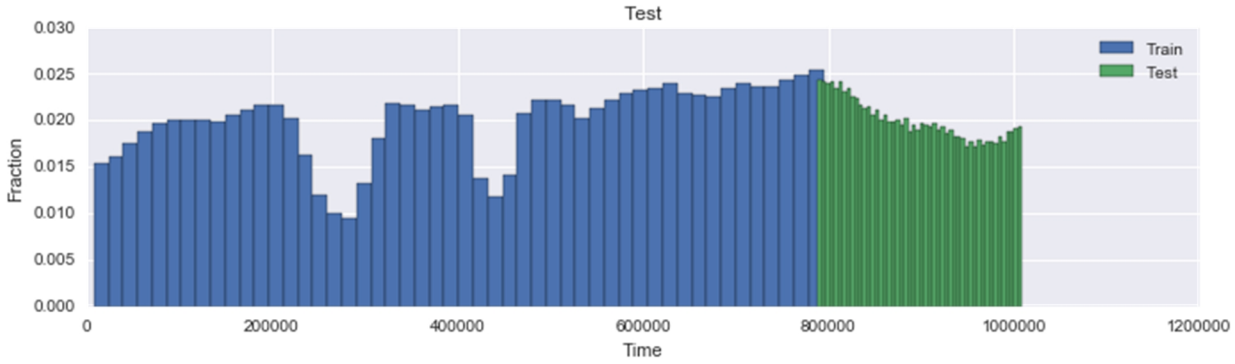
| | x | y | accuracy | time | place_id |
|---|---|---|---|---|---|
| **count** | 8607230.000000 | 8607230.000000 | 8607230.000000 | 8607230.000000 | 8607230 |
| **mean** | 4.991417 | 5.006705 | 92.652076 | 890463.661617 | -1 |
| **std** | 2.866409 | 2.886888 | 124.290613 | 64467.829800 | 0 |
| **min** | 0.000000 | 0.000000 | 1.000000 | 786242.000000 | -1 |
| **25%** | 2.517000 | 2.502400 | 42.000000 | 833220.000000 | -1 |
| **50%** | 4.988000 | 5.000900 | 64.000000 | 887462.000000 | -1 |
| **75%** | 7.463600 | 7.505300 | 79.000000 | 945491.000000 | -1 |
| **max** | 10.000000 | 10.000000 | 1026.000000 | 1006589.000000 | -1 |

Notice that time is given to us simply as a numeric value.

Kyu Cho
5/28/2016

Check time distributions:

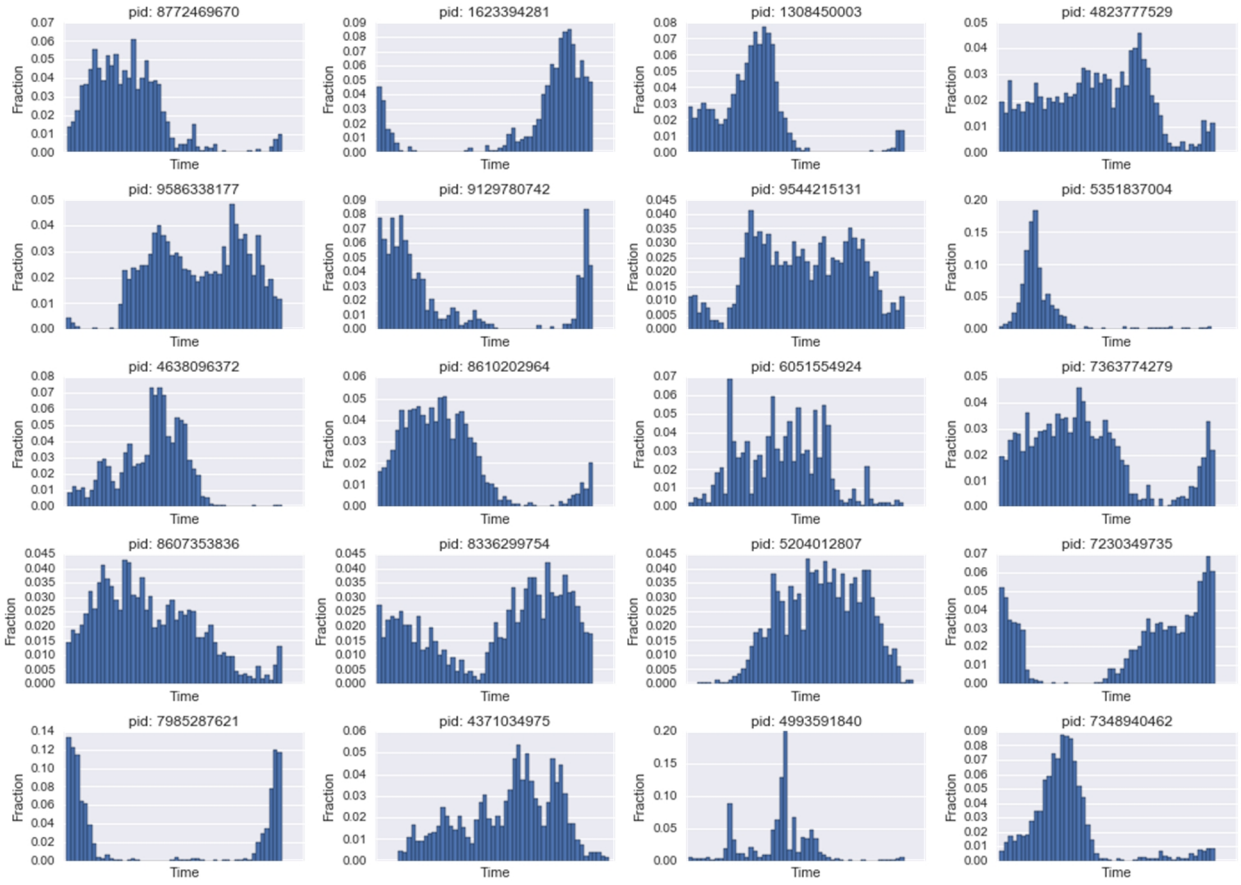


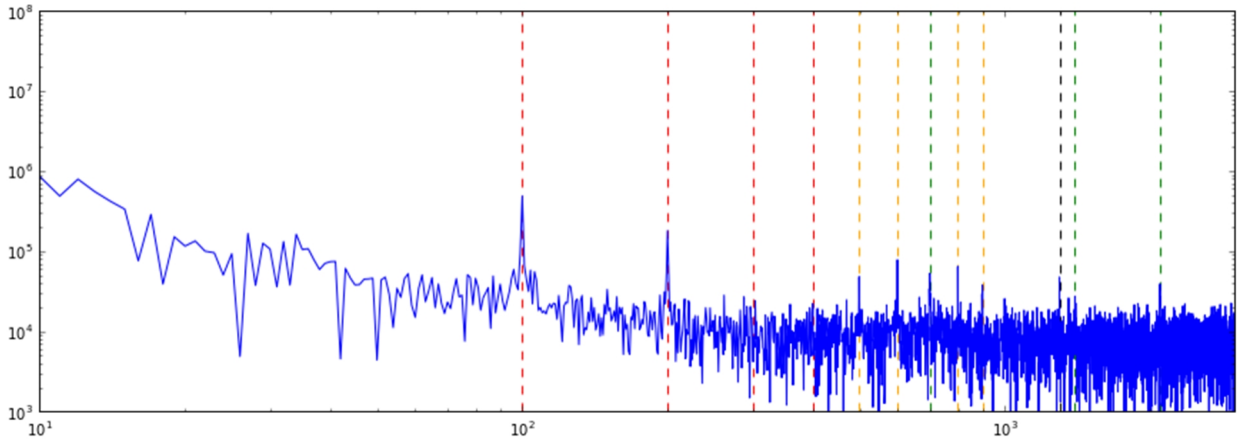

That's interesting: there are two very big drops.

The two dips of time in training set are curious, if looking at counts per unit time they might need to be normalized.

Kyu Cho
5/28/2016




Minutes looks pretty promising.
This means we have ~555 days in train and ~140 in test.
From this, we can look at day of week to identify trends (weekends), day (to find longer term seasonality).
But let's dive into a bit deeper using histogram.

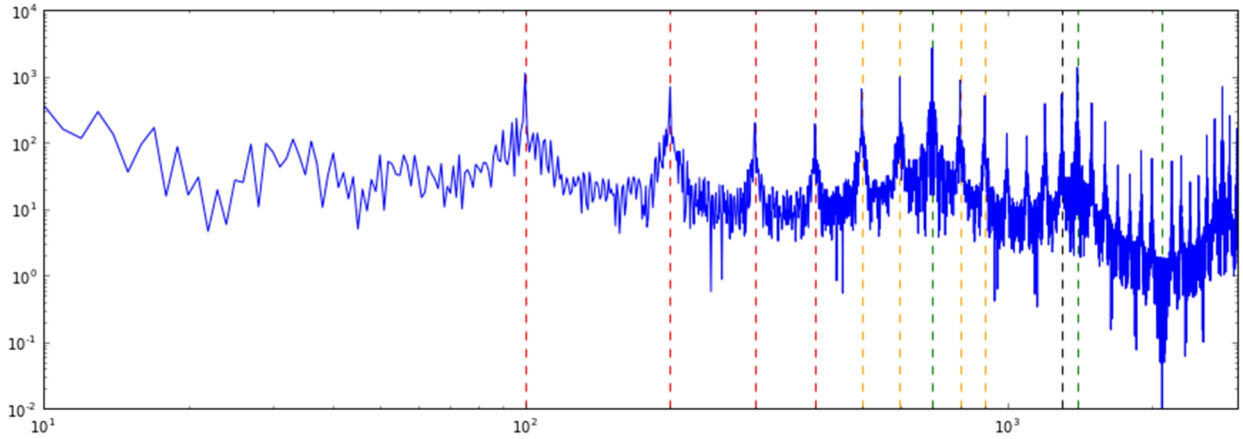


Kyu Cho
5/28/2016

The first peak at the lef (Dashed red line) shows 100 events.
The other redlines are its harmonics.
Interestingly, We do not have any harmonic at 300 and 400.
If we have 100 weeks, then we should have 700 days.
If we zoom in, we see that there is a peak at 699 and its harmonics at 1398 and 2097 (green lines).
Orange lines are modulations of the higher frequency event (days) with a lower frequency event (weeks), at 499, 599, 799, 899.

**Simulation**

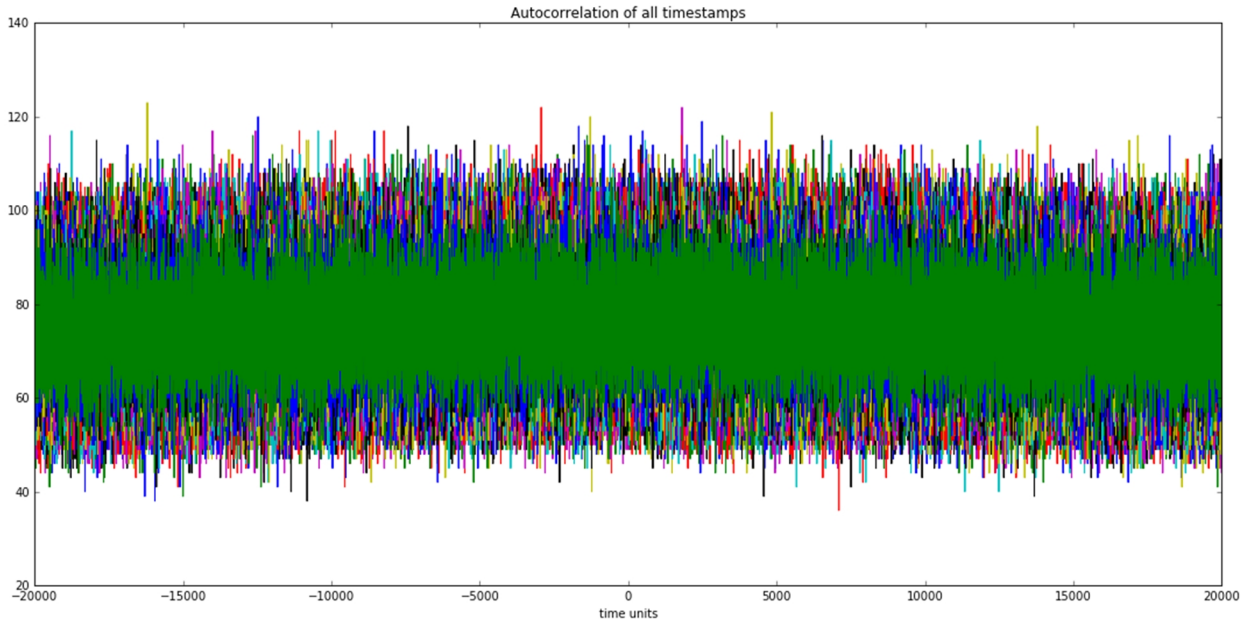
Now lets simulate. We assume a business that is open 8 hours a day, 5 days per week. This business has 1 hour of lunch break. We produce the data for about 100 weeks (699 days)

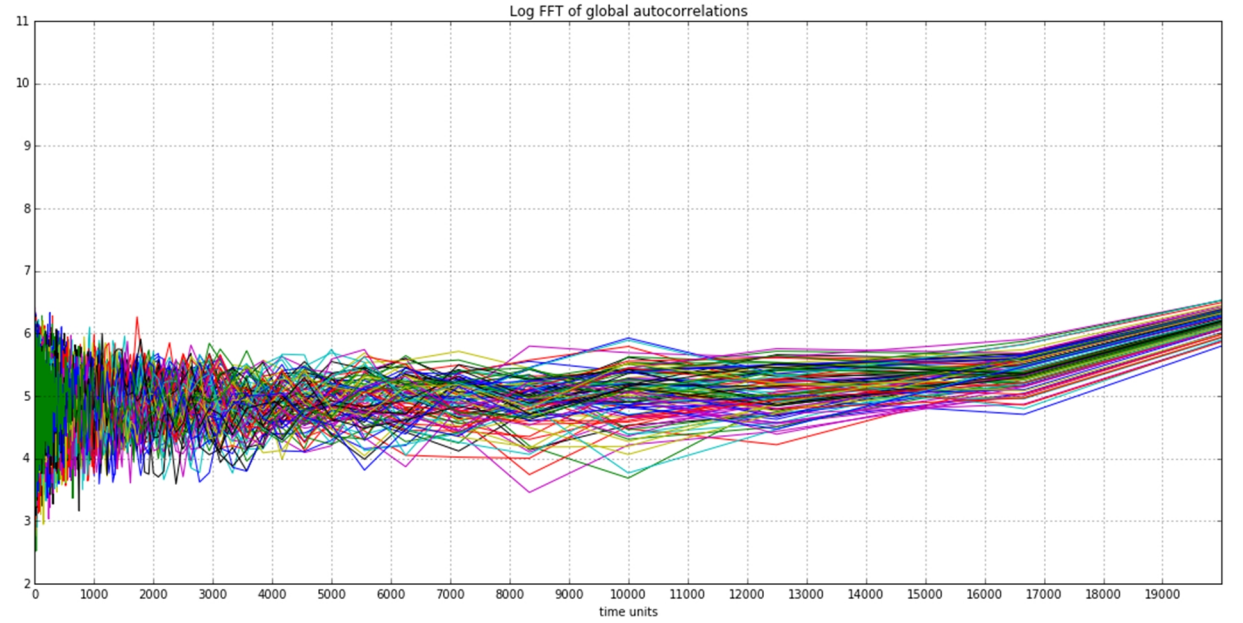



It is similar, isn't it?
My conclusion about time:
We have data for 699 days.
The time step is one minute.
What's also interesting is how evenly dispersed the time is - if this were, say, daily data we'd expect much more of a seasonal pattern.

Kyu Cho
5/28/2016

Let's find if there are really the seasonal pattern exist.

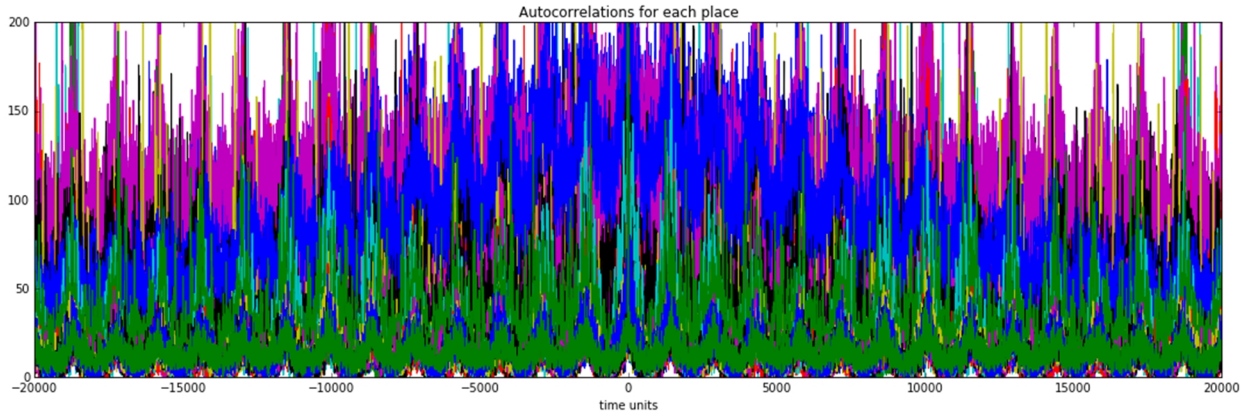


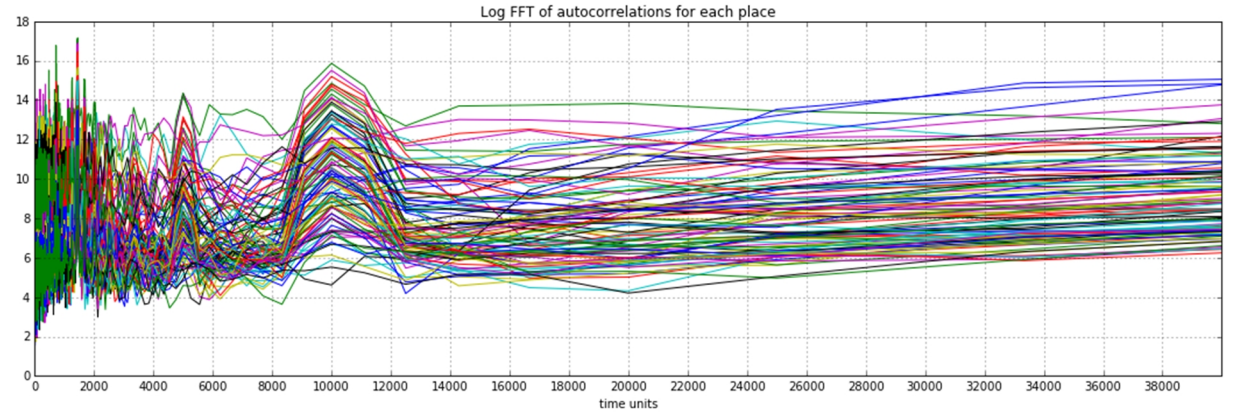



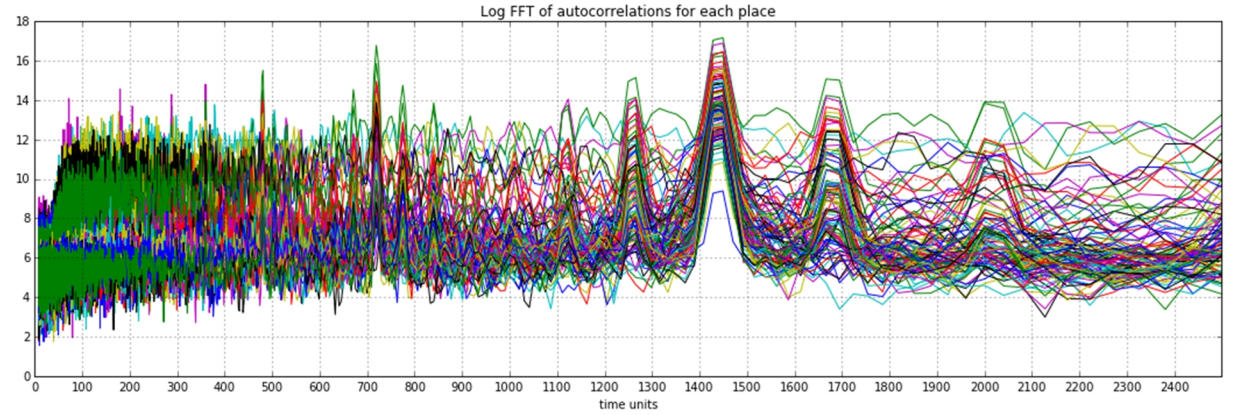

Each of the lines in these plots represent an iteration of random samples.
The top plot is the autocorrelation. Any peaks or waves in the autocorrelation would indicate some kind of repeating pattern in the data.
The bottom plot is created from the FFT of the autocorrelation, which gives the power spectral density (PSD).
Note that the x axis is the inverse of frequency, so it corresponds to raw time units. There are no clear peaks in the PSD.

Kyu Cho
5/28/2016

There does not appear to be any global structure to the timestamps. Maybe there is for individual places?



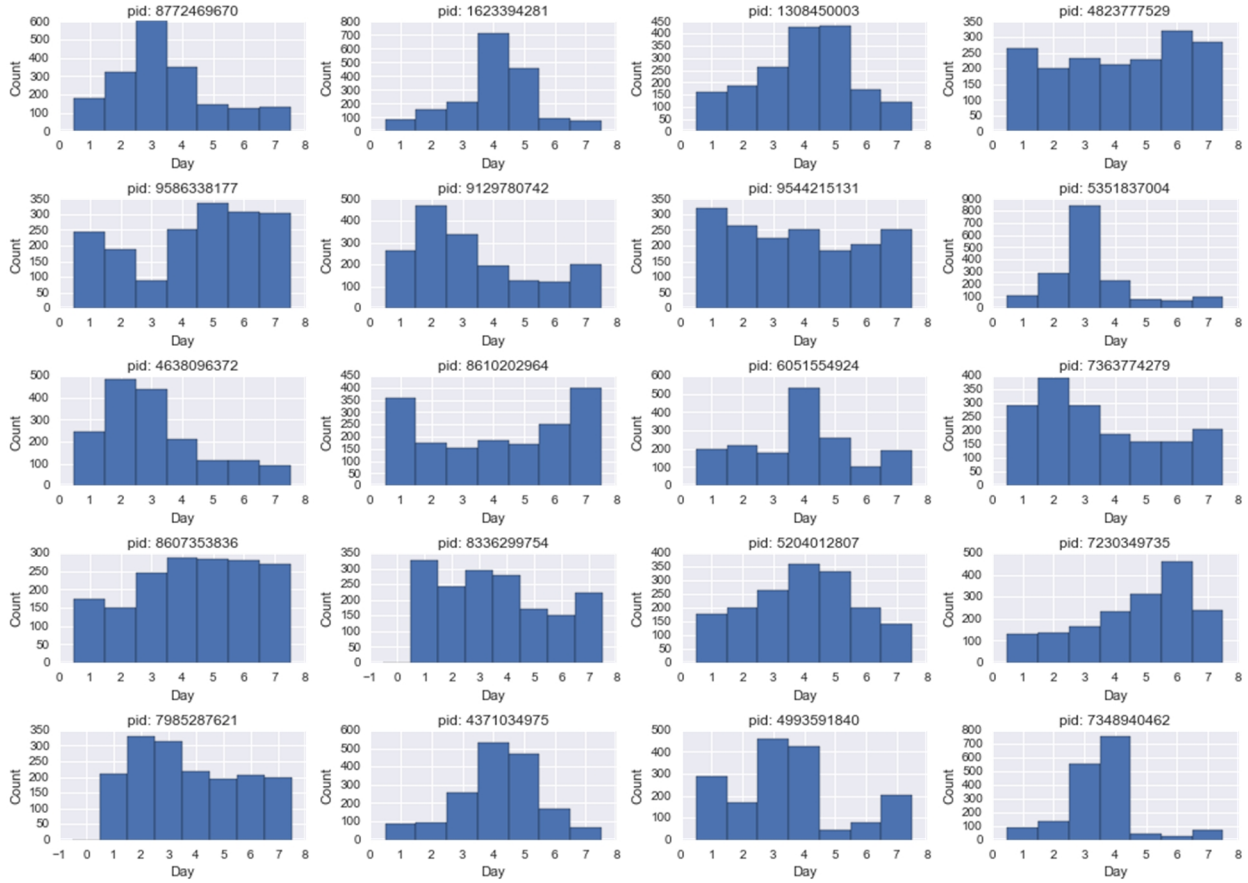




That's more like it!
There is clearly a repeating pattern to this data. A wavelike pattern is seen in the autocorrelations and there are clear peaks in the PSD.
The PSDs give some insight into the different time cycles occurring at each place.
This result confirms that the time units are in minutes. The largest peak is at around 1440, which is the number of minutes in a day. There is another peak at around 10000, which is near the number of

minutes in a week. Some other peaks can be seen too, at 5000, 1650, etc. Not sure what the significance of those are.
The most interesting result is that each place kind of lives in its own little time zone. You'd think that the people living in this world would all behave according to a particular cycle, but that doesn't pan out in the global timestamp data.

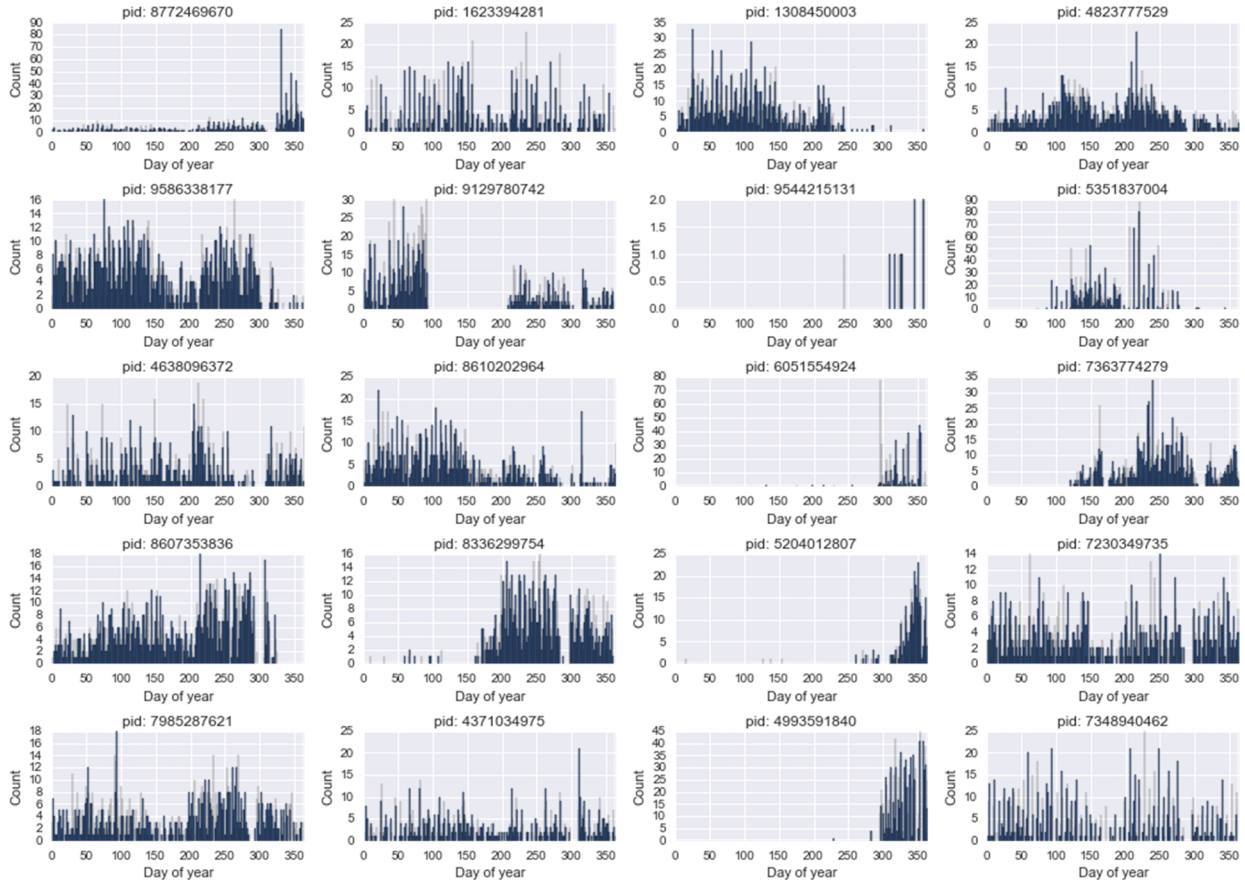
The next step is to add some columns representing our new time.
Getting this exactly right (within the minute, so that "hours" are defined by clock hours, which probably correlate better with place visits) will probably be crucial later on, but as a first it doesn't matter if we're out by a bit.

Look at the aggregate number of visits per weekday for the top 20 locations, this should show weekends, hopefully.




Some appear to have weekend-like behavior... what about looking at day of year.

Kyu Cho
5/28/2016

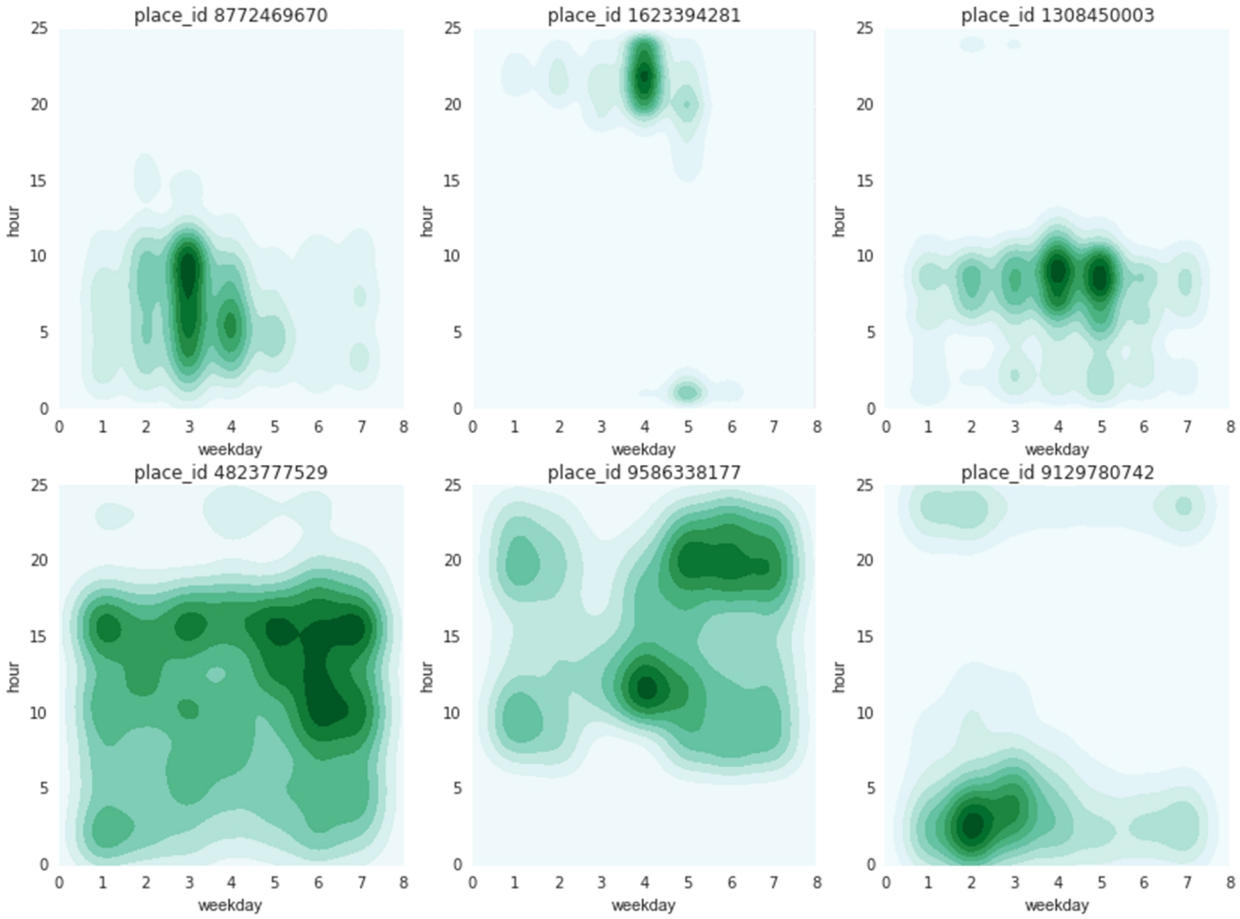



Somewhat weird trends, looks like some of the top places only opened up business part way through.

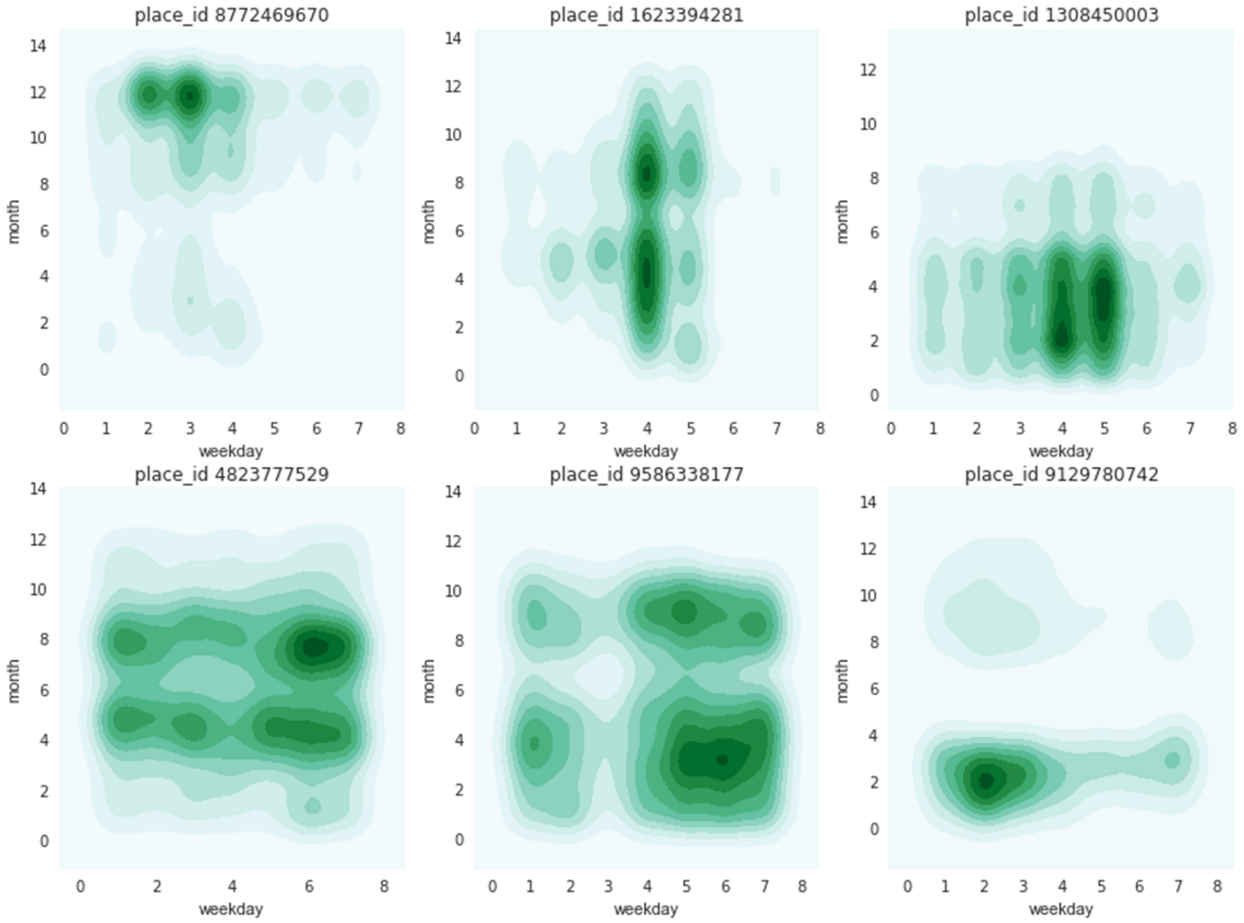
The plots show a preference for certain hours and weekdays for each place id.

Kyu Cho
5/28/2016

Maybe weekends and weekdays and holidays can be separated if more place ids are analyzed.



Kyu Cho
5/28/2016




We can see the month to month changes in the above plots for each place id.

Some businesses seem to be highly seasonal.

Next, let's plot a very small subset of train and test data and compare the kdes.

This is just a confirmation that the test data is lining up well with the train data.

Kyu Cho
5/28/2016

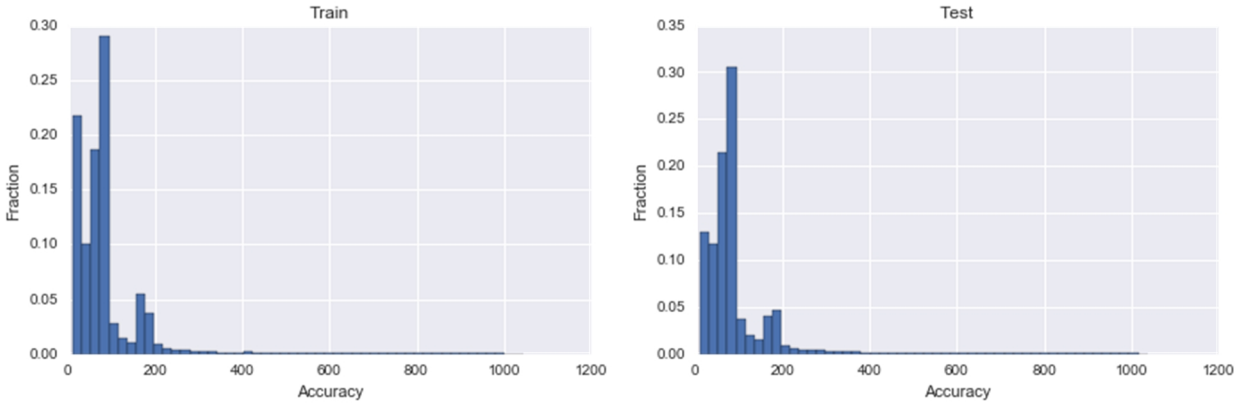
# Accuracy

We already know that accuracy isn't exactly defined. From first principles, we could think of it a few ways -

- Error on some arbitrary scale. This seems unlikely, since the max is > 1,000.
- Error on the same scale as x and y. Now, this could be an estimated radius (with the x and y values as the center); either normal or squared.

Since we have a lot of data, and we're running this in Kaggle scripts, we can randomly sample 1% of the data and look at the data. The pattern will (almost certainly) be the same.

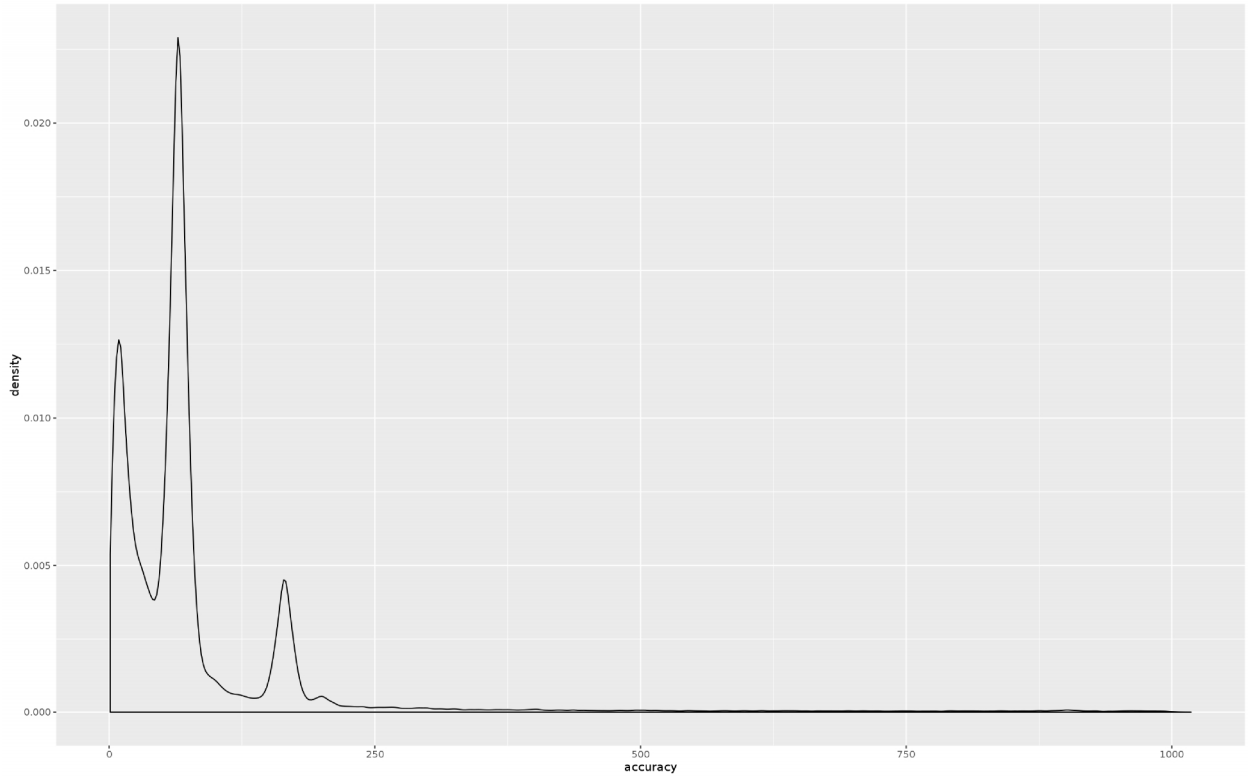
Let's start with some basic histograms, showing the distribution of accuracy and time.




Accuracy has some interesting structure, but is relatively consistent across train/test.

Kyu Cho
5/28/2016

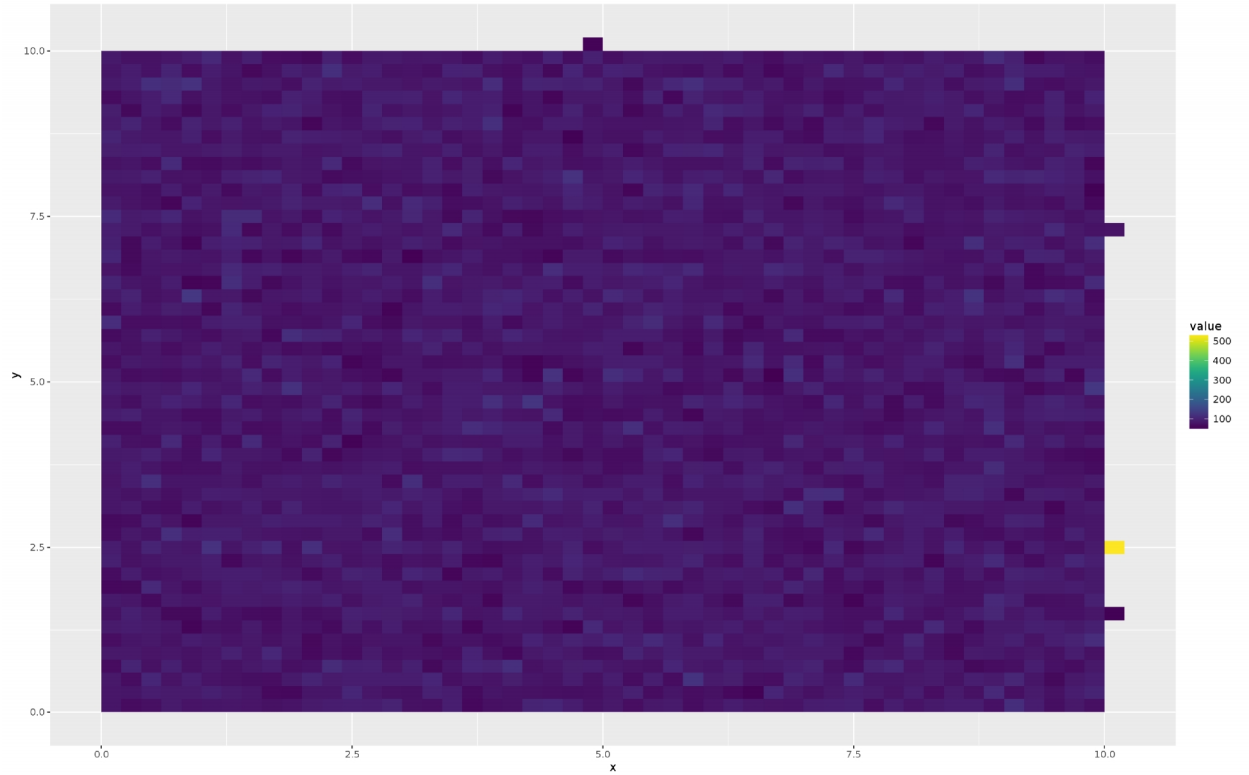
Let’s look at density plot for accuracy.




It looks like we have three peaks.
Do we think there are underlying parameters in the simulation at these peaks?
We might also think that there are different measures of accuracy at different locations.

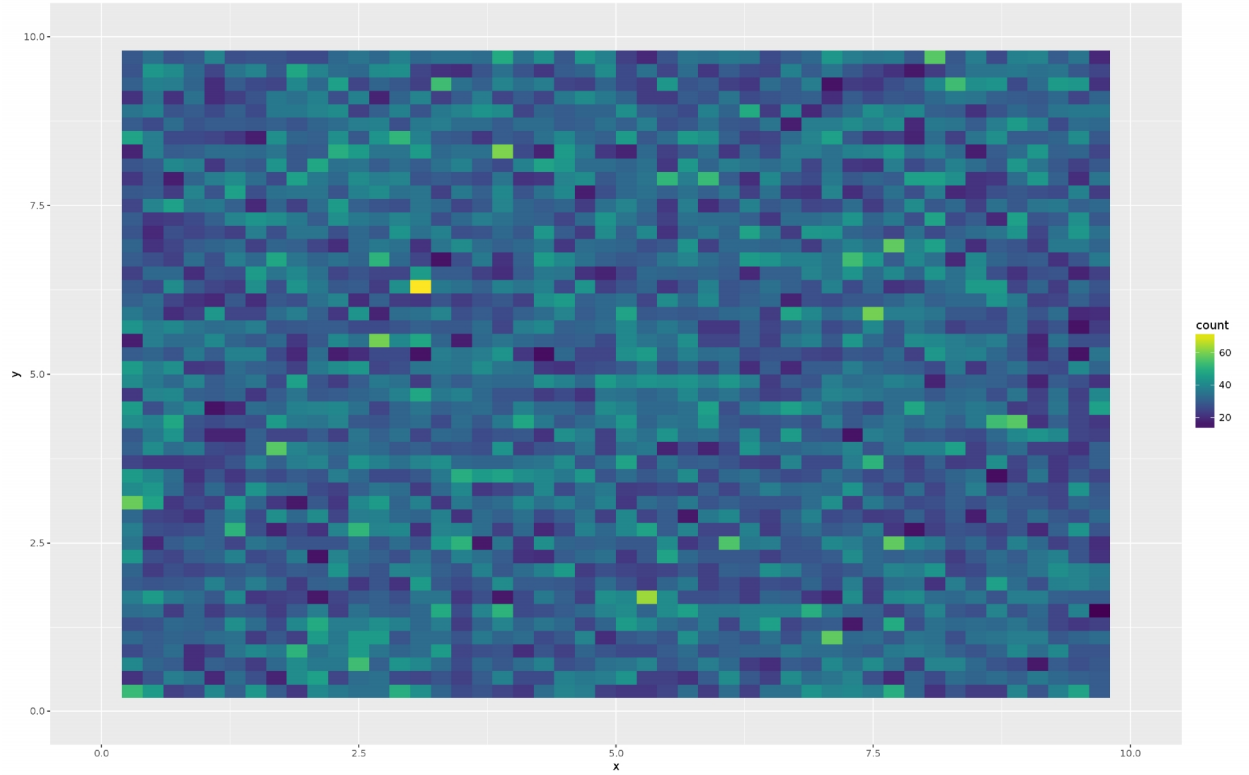
Kyu Cho
5/28/2016

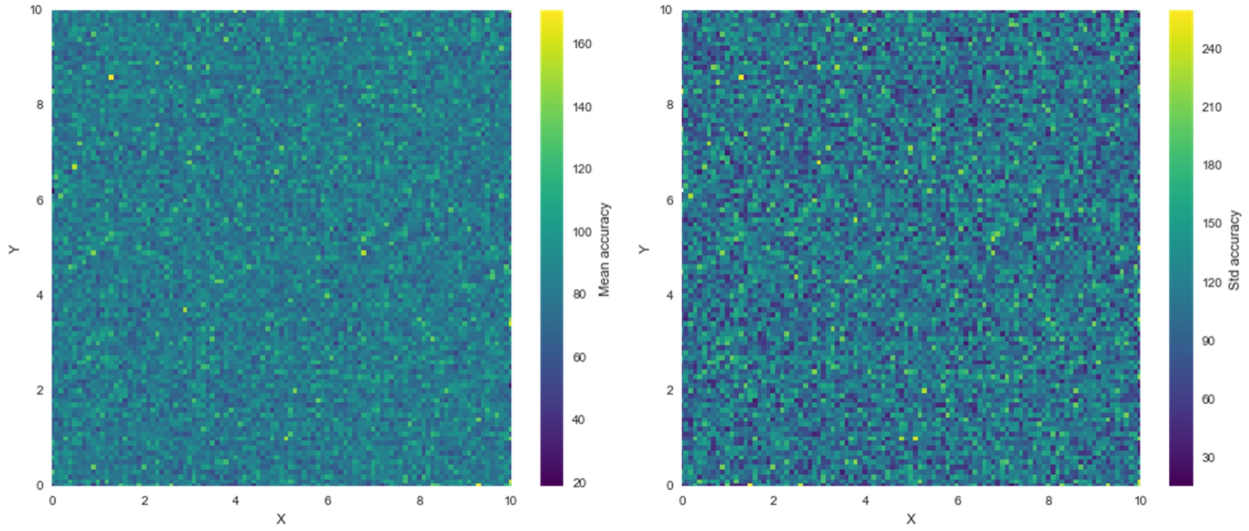
Let's see how even the accuracy is over x and y.
Since we have two dimensions and quite a few buckets (50 x 50 = 2,500 bins).




Now, that looks pretty random to me. Doesn't look like there are either (large) hotspots, or a tendency across the whole square.
We can look at it a few other ways (median, max, min, sd, var, etc), but I don't think they add a great deal.
Even if the general distribution is pretty even, are the high-accuracy values (which we think mean low accuracy) anywhere in particular?

Kyu Cho
5/28/2016

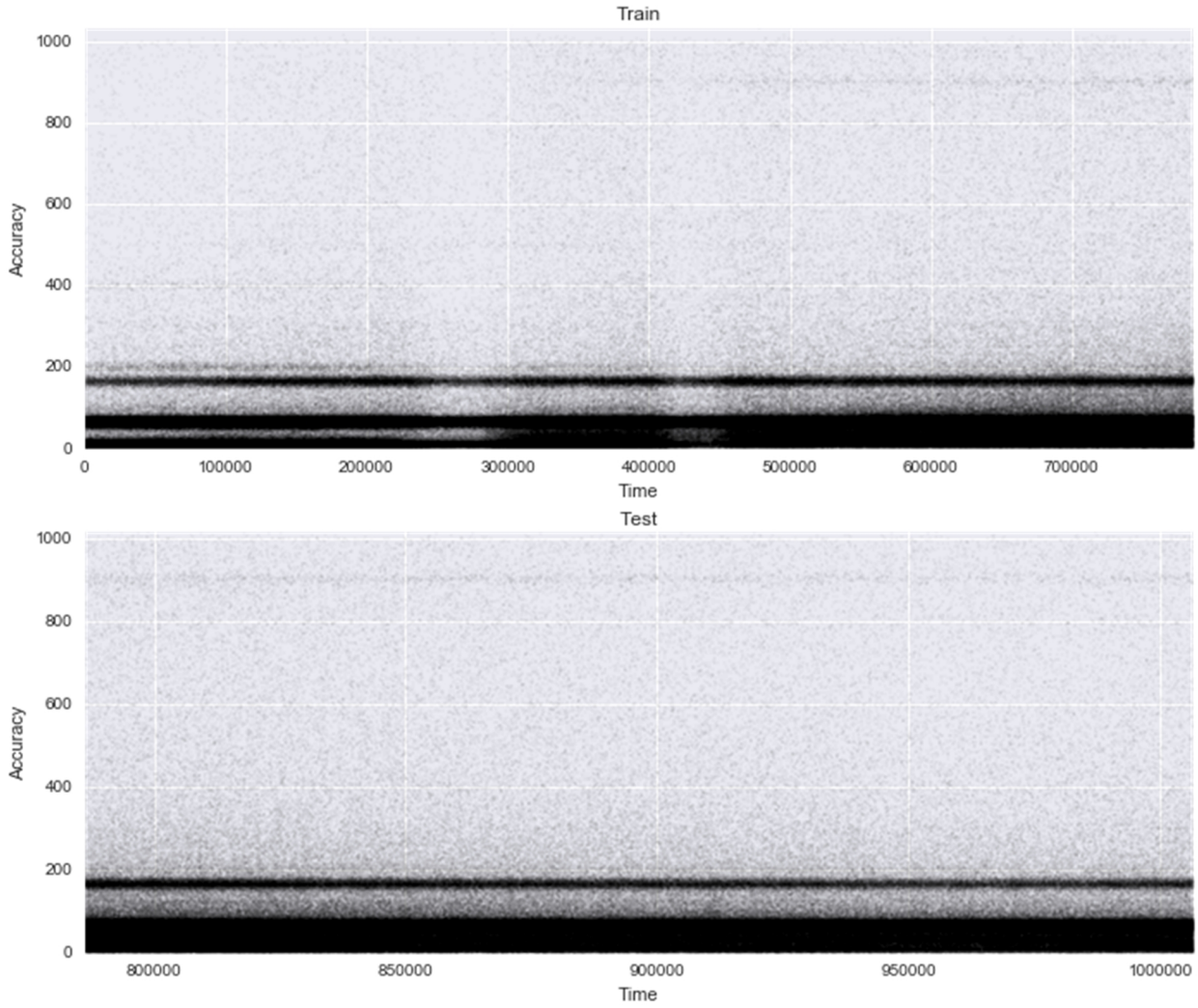



I guess the short answer is “not really”. If there’s any signal in there, it’s not popping to eyes.

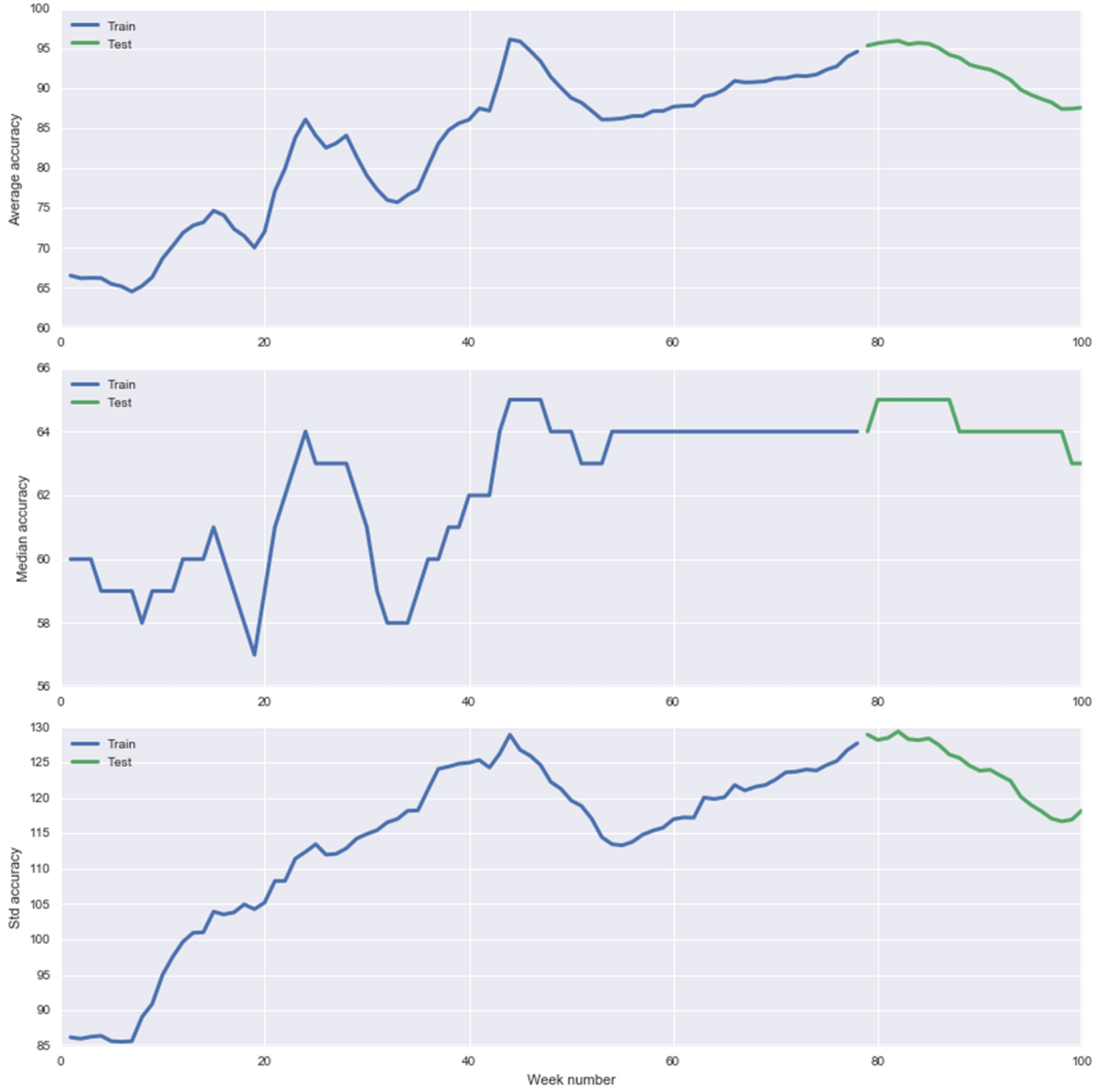
Kyu Cho
5/28/2016

Let's see if accuracy changes with "time" at all:



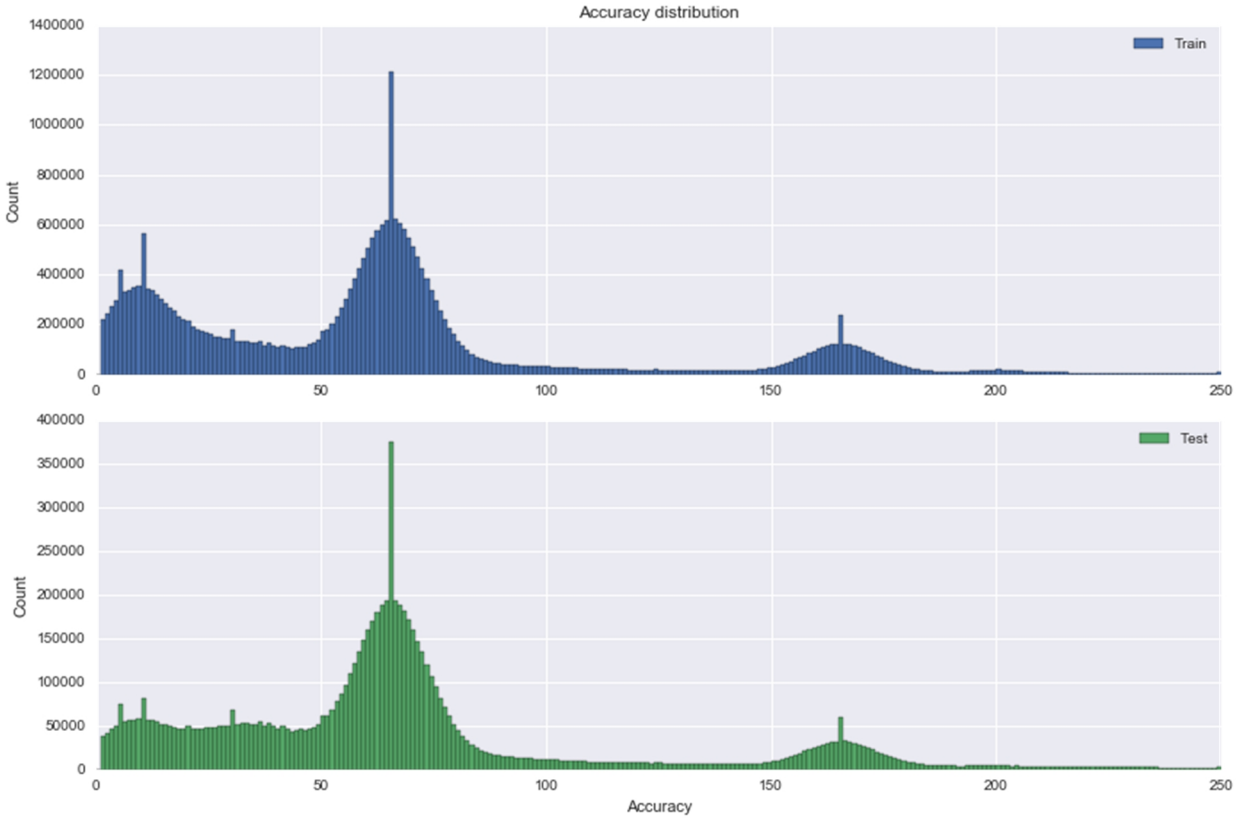

What if the accuracy varies as a function of time?
Not really - but we can see the two time dips in the training plot, and this emphases that accuracy is somewhat preferentially banded.

Kyu Cho
5/28/2016



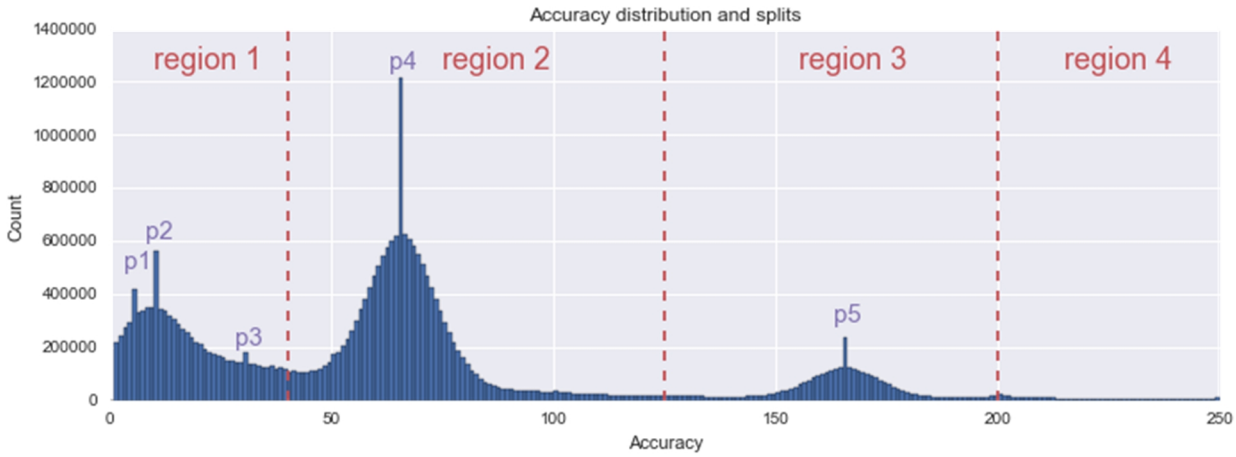

It's very interesting that after ~50 weeks the median accuracy flat lines at 64, with a small blip at 65, then back to 64 for the test data.

Kyu Cho
5/28/2016




We can also examine the distribution of accuracy closer, and find some interesting things
There's three main peaks, as well as 5 weird peaky outliers, let's find out what values they are at:

('Peaks at:', 5.0, 10.0, 30.0, 65.0, 165.0)
('Counts are:', 416462, 561614, 178786, 1212970, 237762)
We can define these regions and peaks, so we can investigate them independently to see if they behave differently in any ways.

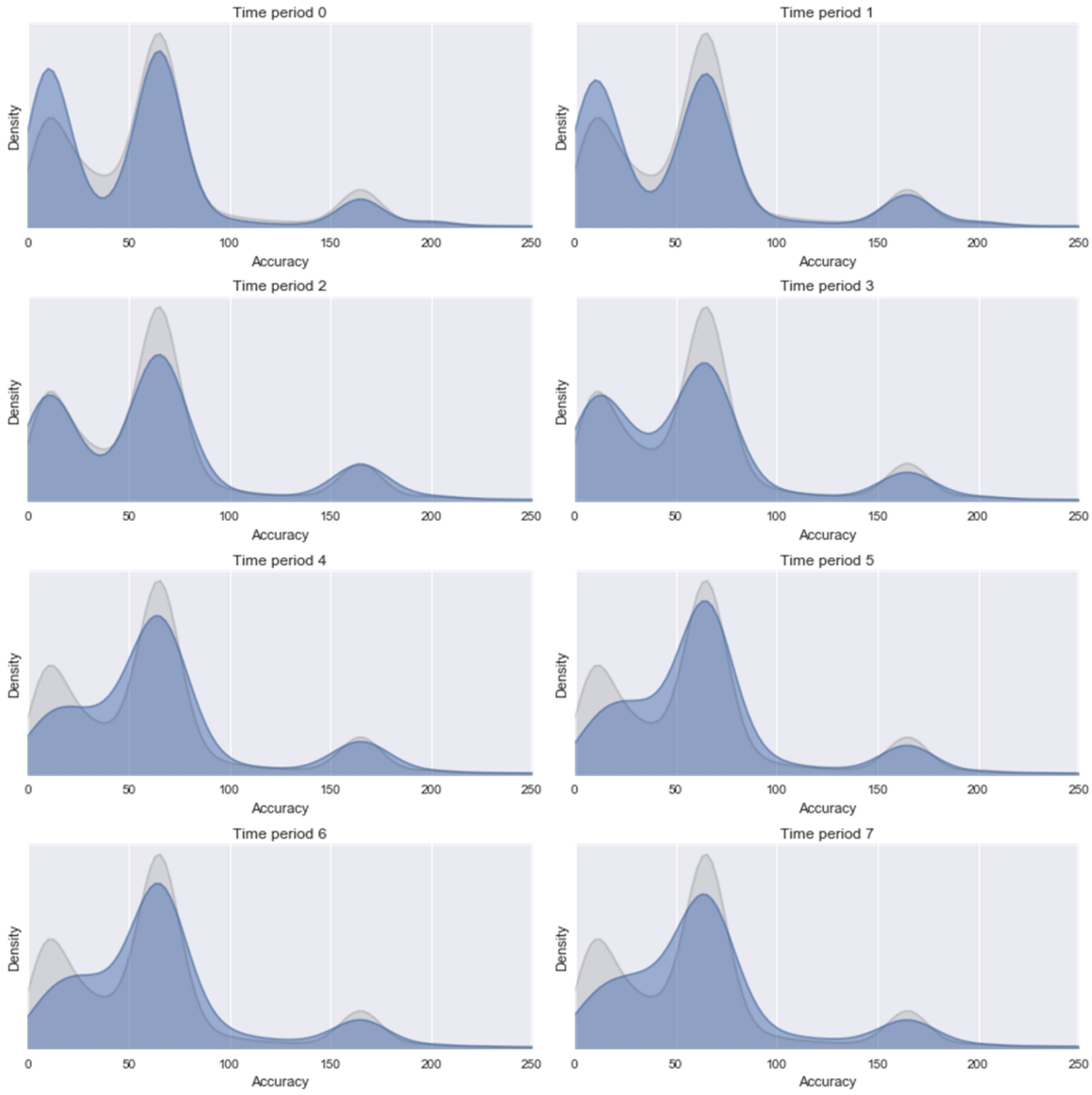


Kyu Cho
5/28/2016

The question is, do these different region/peaks correlate with anything else?
Let's split the train set up into various subsets representing these regions/peaks, and see if the distribution of other variables change.


Everything remains constant except non-cyclic time.
Particularly interesting is that peaks 1 and 2 (the 5th and 6th rows) don't appear much at the end, but peak 3 (7th row) is the reverse.
Let's take raw time and split it up into various even parts, and see how the accuracy distribution varies.
For this, it might be nicer to plot density plots, and compare them against the overall distribution (represented in the following charts as the light grey shaded area).

Kyu Cho
5/28/2016



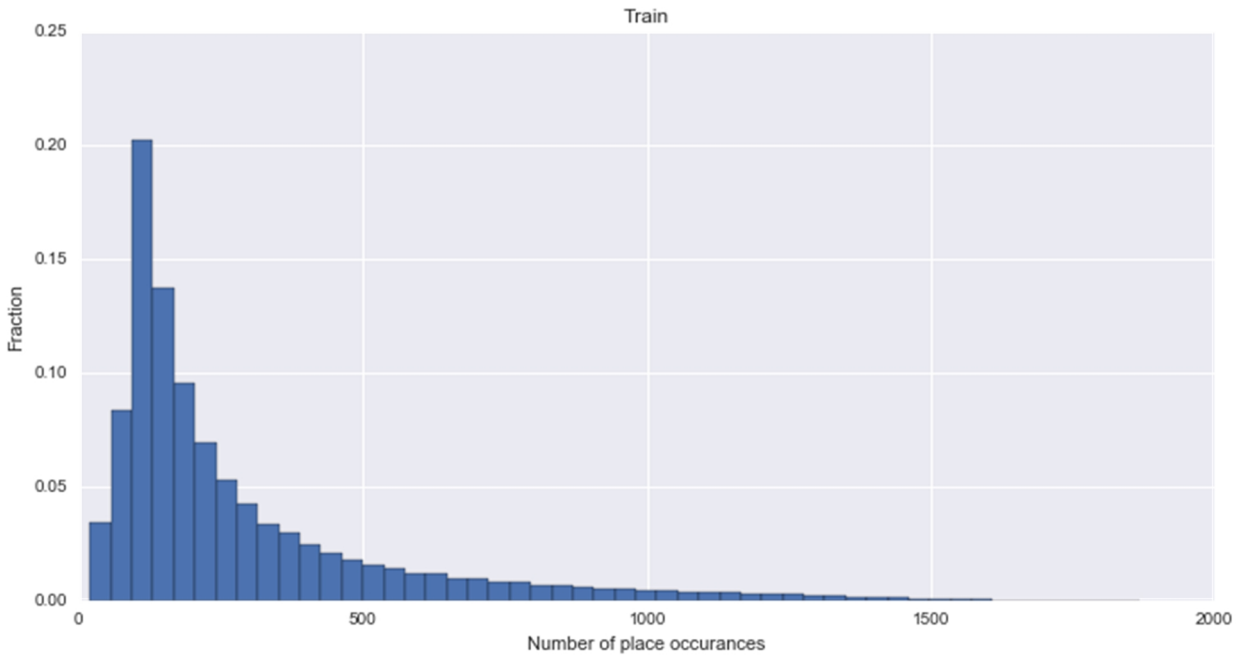

This aligns with the general statement "accuracy increases with time", but actually shows how the distribution changes with time.
You see the biggest change is that as time progresses, region 1 (as defined above) reduces in size and the values move upward. What does this mean though?
Knowing accuracy distribution as a function of time (in this form) is useless for predictions...

Kyu Cho
5/28/2016

# Location

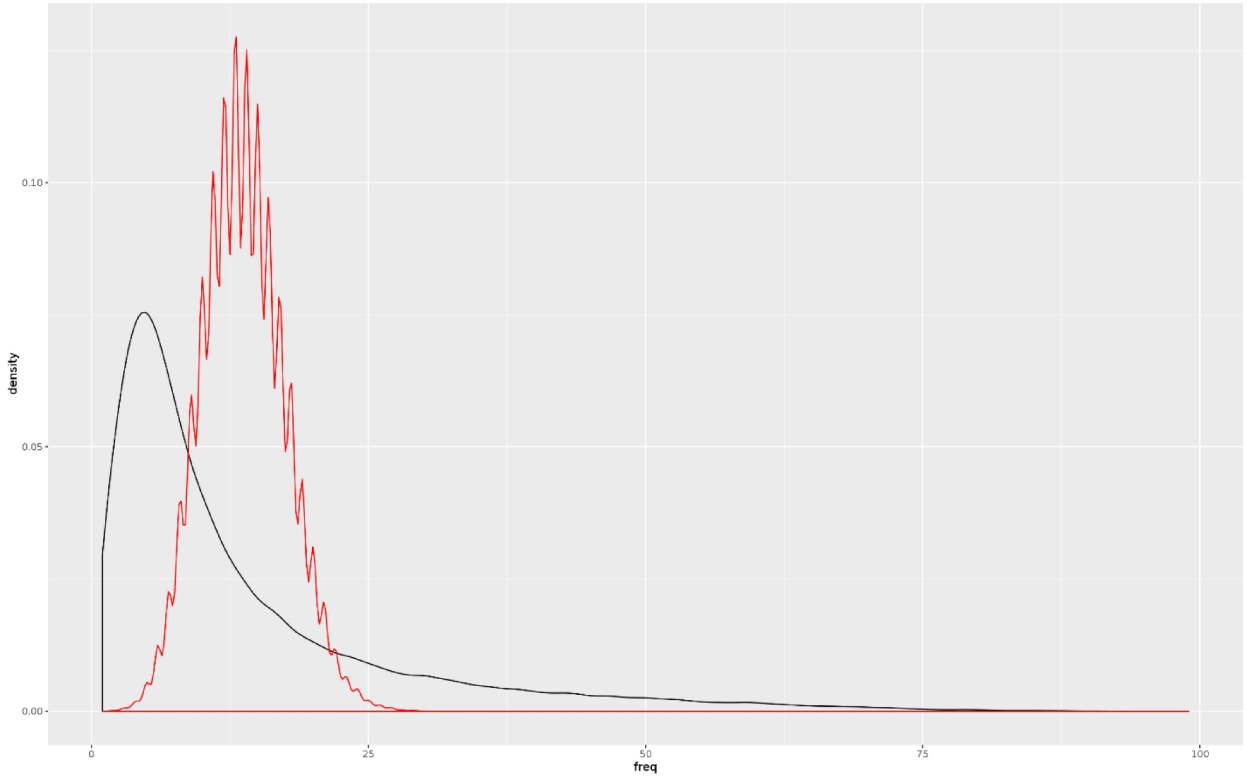
| | y | x | time |
|---|---|---|---|
| **place_id** | | | |
| **8772469670** | 0.011544 | 0.356140 | 1849 |
| **1623394281** | 0.015712 | 0.240642 | 1802 |
| **1308450003** | 0.017662 | 0.340147 | 1757 |
| **4823777529** | 0.011860 | 0.476614 | 1738 |
| **9586338177** | 0.014817 | 0.507308 | 1718 |

We can look at is how frequently different locations appear.

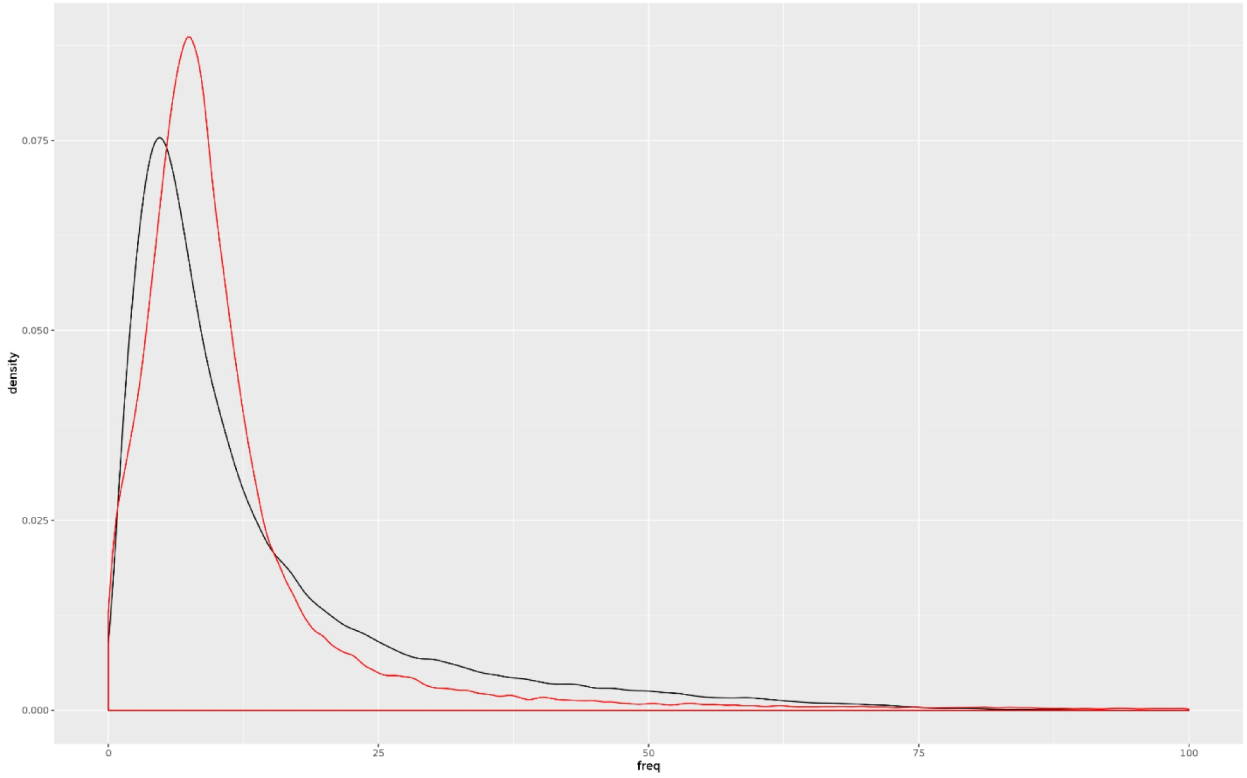



OK, so most places appear around 100 times.
Let's look at a bit deeper.
Well, if that doesn't look like a Poisson distribution!
Let's fit and compare.

Kyu Cho
5/28/2016




Well, obviously, it's not. Probably something bounded below by zero with a much fatter tail. Perhaps a Cauchy distribution?



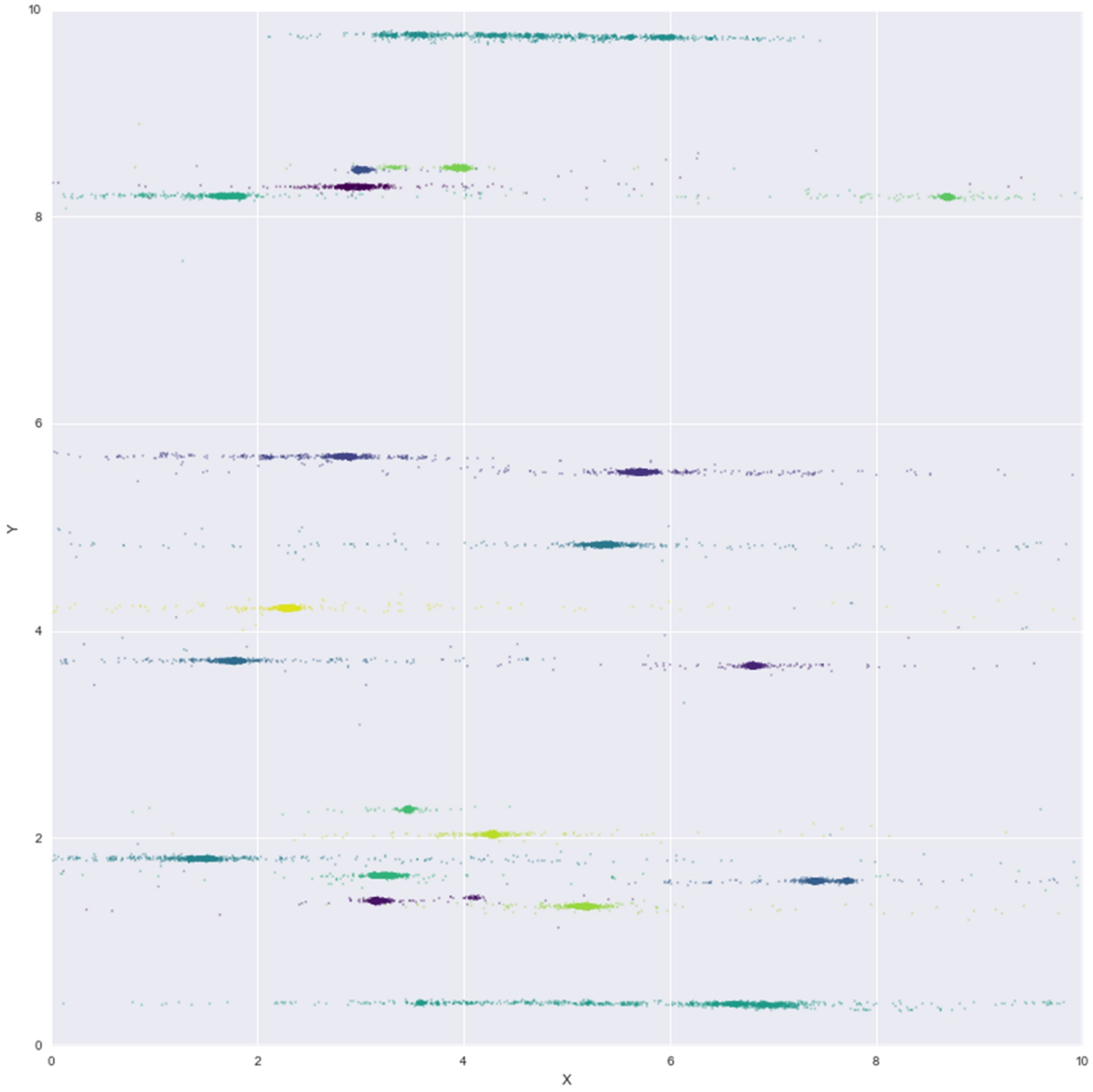
Kyu Cho
5/28/2016

Well, it's better.

We could play around with this for a while; the principle point is that this looks like a pretty clean distribution for frequency, which is nice. Humps - or something else suggesting another type of mixture - would be far harder to work with.

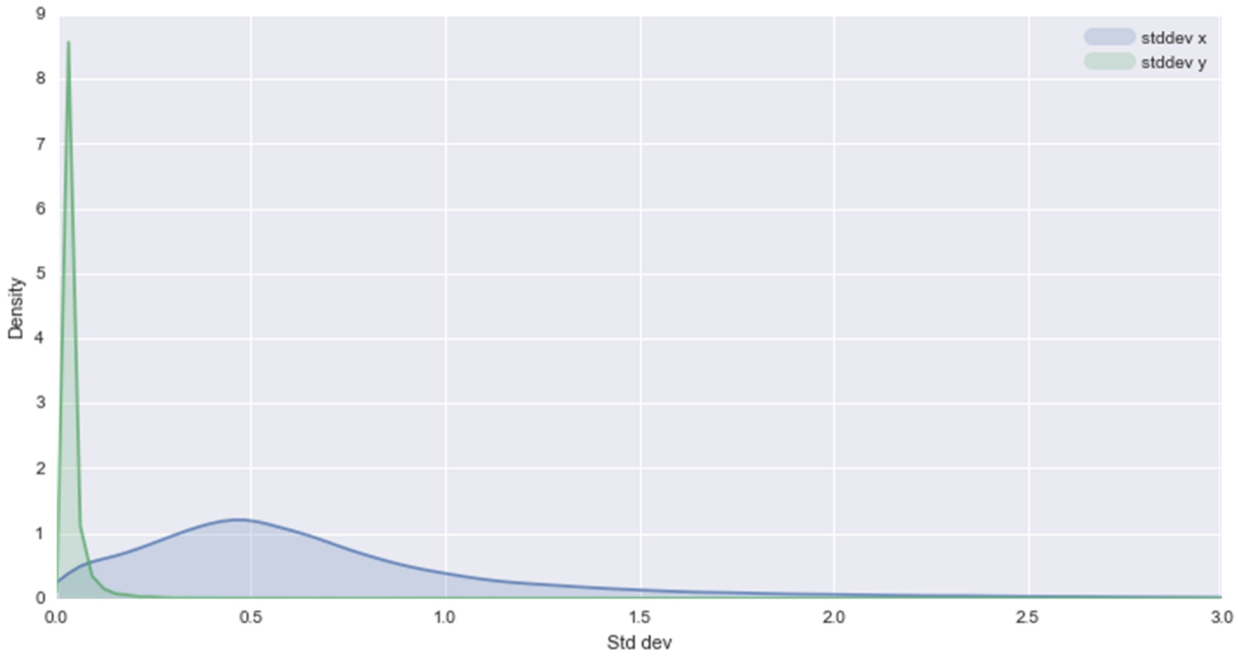
How variable are the estimates for the popular places?

Let's take a handful (10 places) that have a frequency of more than 10 (in our sample) and find the x and y parameters for each.



Kyu Cho
5/28/2016

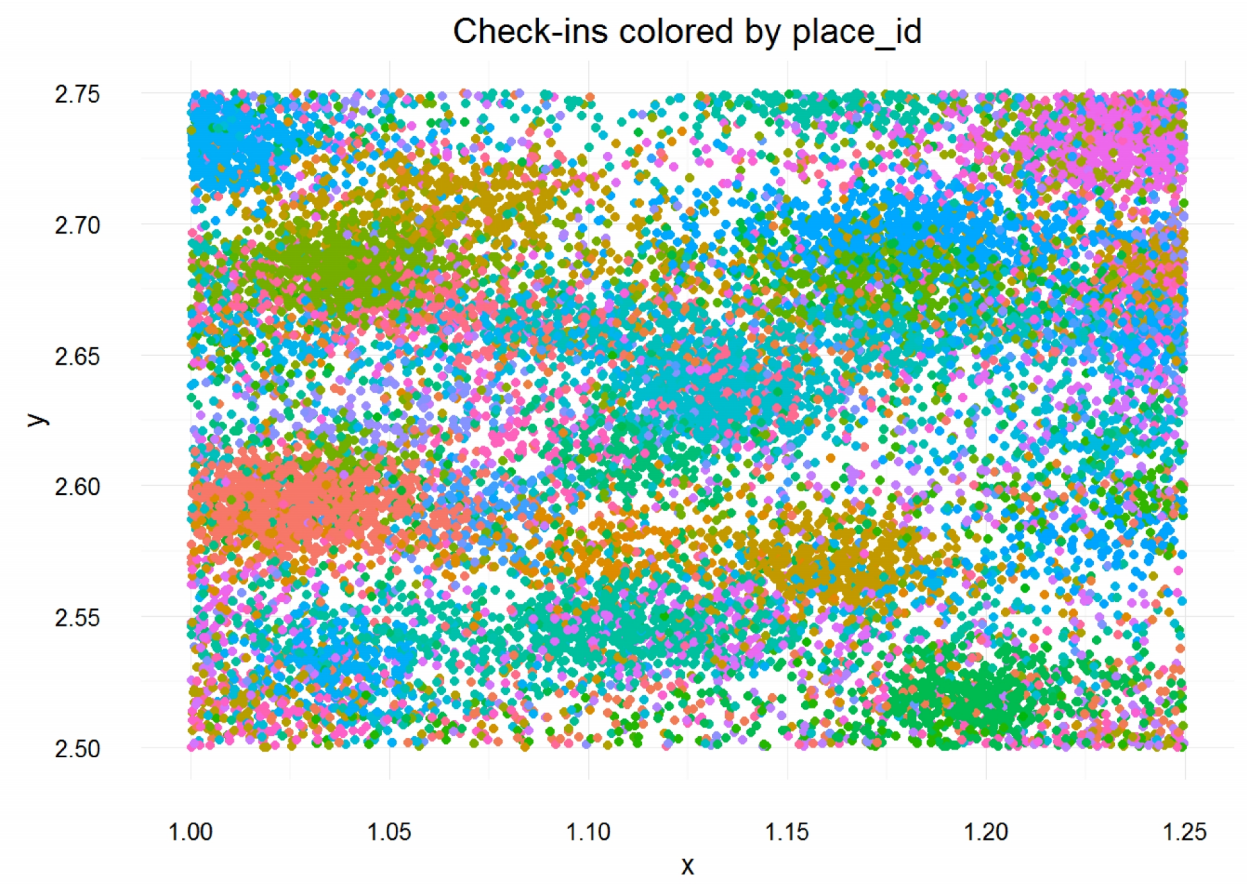
That's interesting. While in general it seems like the places with the small accuracy scores are in the middle, there are some pretty odd patterns here.
This highlights the variance, looking suspiciously like streets. We can look at the standard deviations to illustrate this:




| place_id | mean_x | sd_x | mean_y | sd_y |
|---|---|---|---|---|
| 8296848516 | 8.12 | 0.94 | 2.63 | 0.01 |
| 1978057520 | 4.68 | 0.86 | 9.30 | 0.02 |
| 5209842509 | 4.86 | 0.64 | 0.27 | 0.01 |
| 1052628400 | 2.91 | 0.27 | 6.26 | 0.01 |
| 4444032048 | 5.73 | 0.18 | 0.68 | 0.01 |
| 7729330444 | 8.86 | 0.15 | 7.90 | 0.01 |
| 4823777529 | 5.70 | 0.09 | 5.53 | 0.01 |
| 9116547026 | 7.28 | 0.07 | 8.35 | 0.01 |
| 3659348746 | 1.41 | 0.06 | 3.34 | 0.03 |
| 7698408658 | 6.49 | 0.04 | 0.47 | 0.01 |

That's such a weird pattern. And note that the biggest y value is 0.04 - the same as the smallest x value. Crazy.

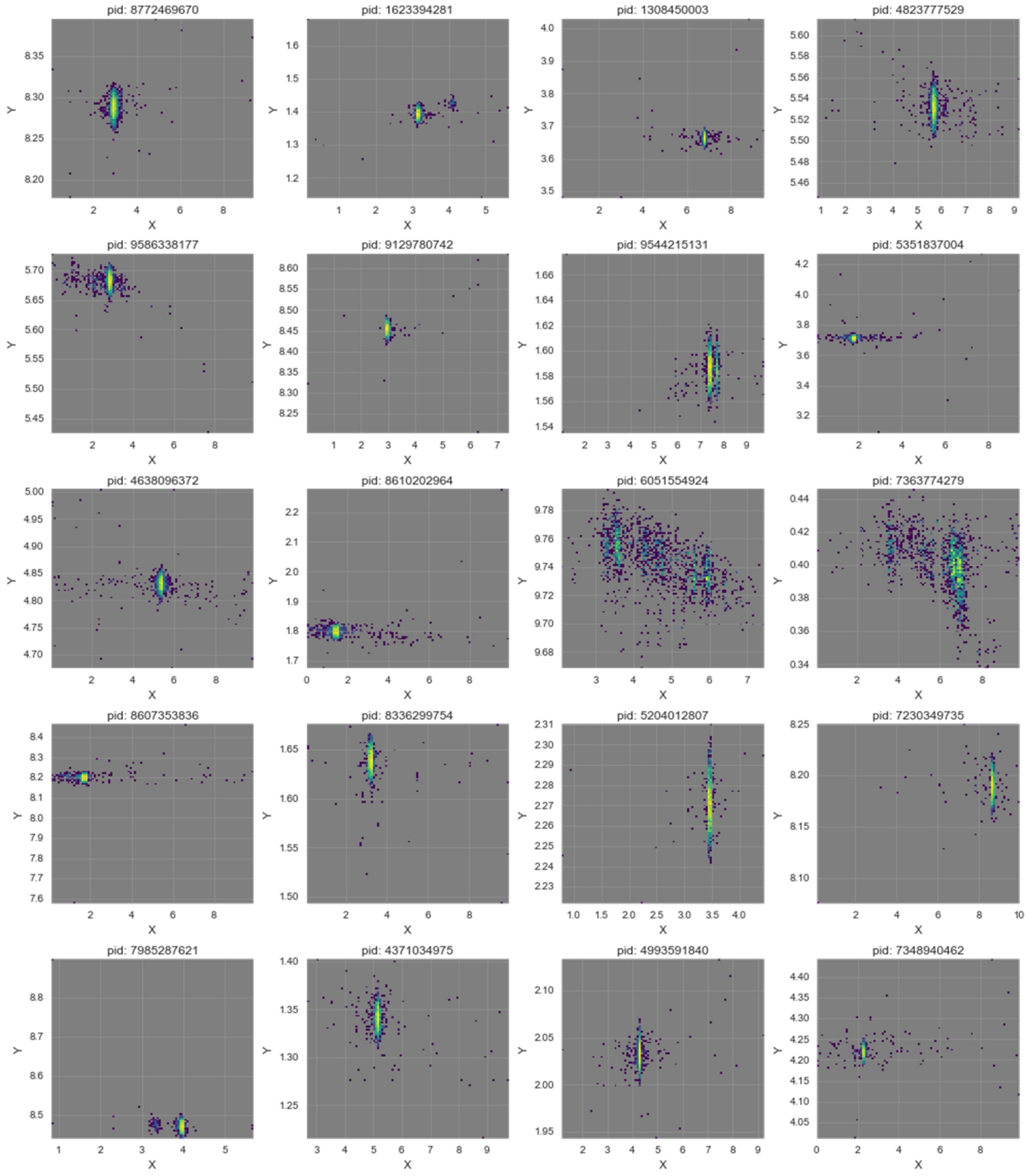
Kyu Cho
5/28/2016




Ok, so the clusters are pretty visible, however there seems to be quite a lot of overlap - the place_id's are definitely not separable.

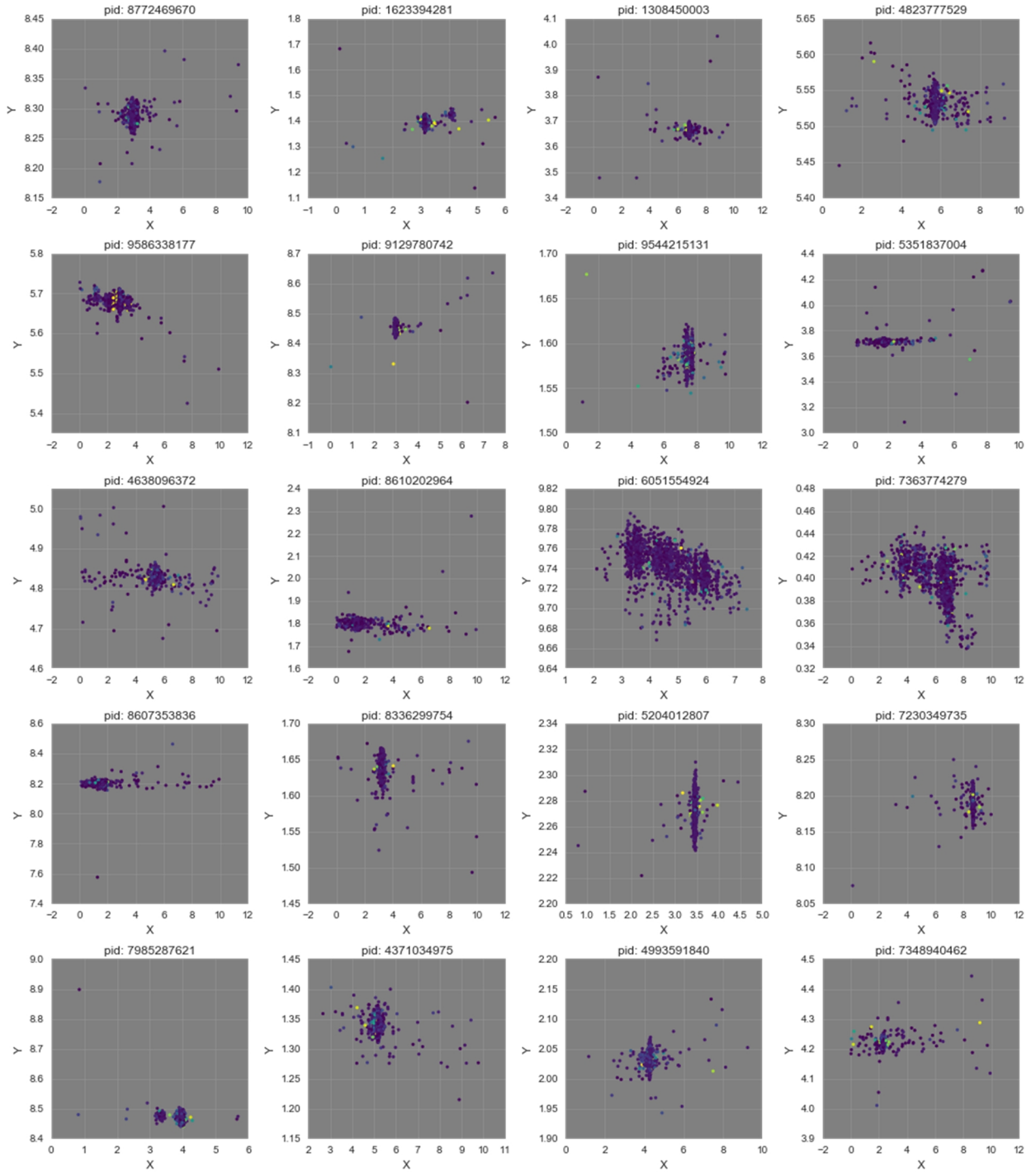
Let's try plotting them using the hour component as our third variable.
We will just look at the most popular clusters otherwise it gets really messy:

Kyu Cho
5/28/2016



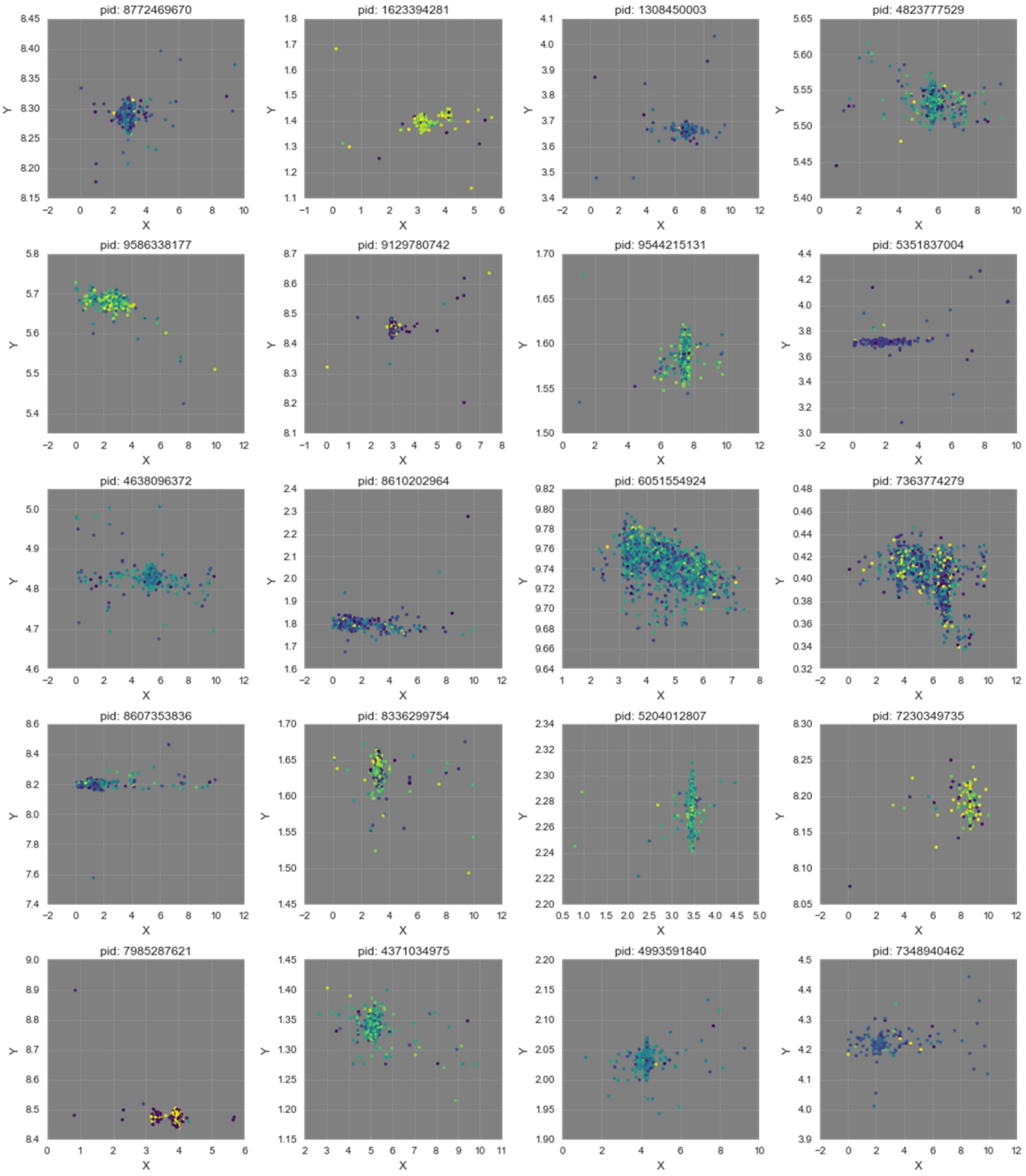

The distributions are different for different locations, but many span a huge x-range relative to the y-range (maybe roads are aligned this way?)
Let's re-visit the accuracy to see if it changes we get further away from the presumed centroid location:

Kyu Cho
5/28/2016



Kyu Cho
5/28/2016

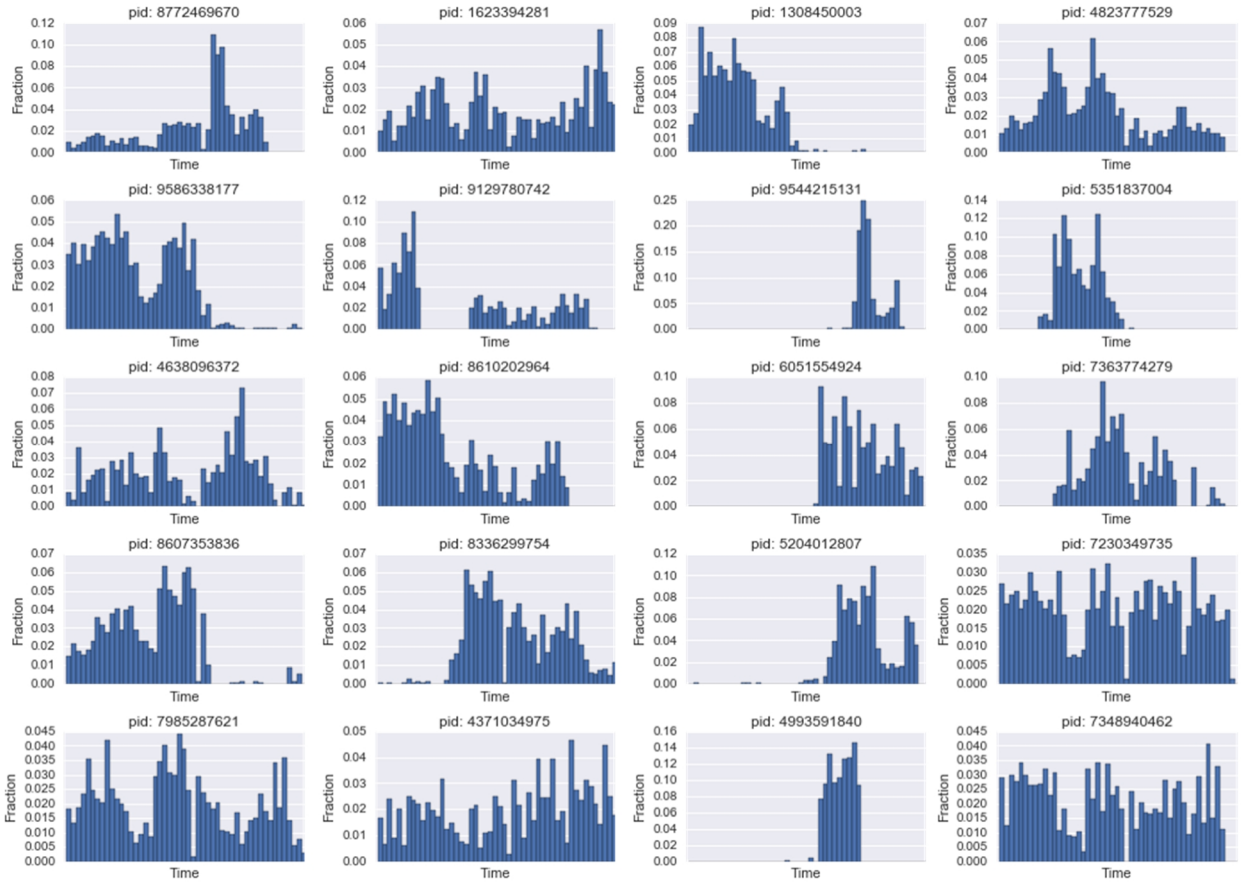
Nope, not really. What about time variance per location?




This certainly shows different places are preferentially visited at different hours.
This will be useful for predictions, since for a given "hour" the list of probably places will be reduced.
Let's pick an arbitrary place, and see if it's shape is discernible over the background noise.

Kyu Cho
5/28/2016

Let's look at the location in function of time in density graph.




The daily cycles are clearly visible above - for certain places the check in's stop for a few hours and then start picking up again.

Other businesses have quite a few peaks throughout the day, and the peaks tend to be rather different for different businesses.

Also keep in mind that the upper z-square (z = 24) and the lower z-square (z = 0) are really the same thing since time of day is, well, periodic.

So really this thing we're looking at is better viewed not as a cube but as a (flat) solid torus!

Kyu Cho
5/28/2016

# The Forest / Conclusion

Let's use the ranger implementation of the random forest algorithm.

The ranger package tends to be significantly faster(around 5x) and more memory efficient than the randomForest implementation and we'll need as much of that as we can get for this problem.

```
set.seed(131L)
small_train$place_id <- as.factor(small_train$place_id) # ranger needs factors for classification
model_rf <- ranger(place_id ~ x + y + accuracy + hour,
                   small_train,
                   num.trees = 100,
                   write.forest = TRUE,
                   importance = "impurity")
```

```
## Growing trees.. Progress: 42%. Estimated remaining time: 42 seconds.
## Growing trees.. Progress: 84%. Estimated remaining time: 11 seconds.
```

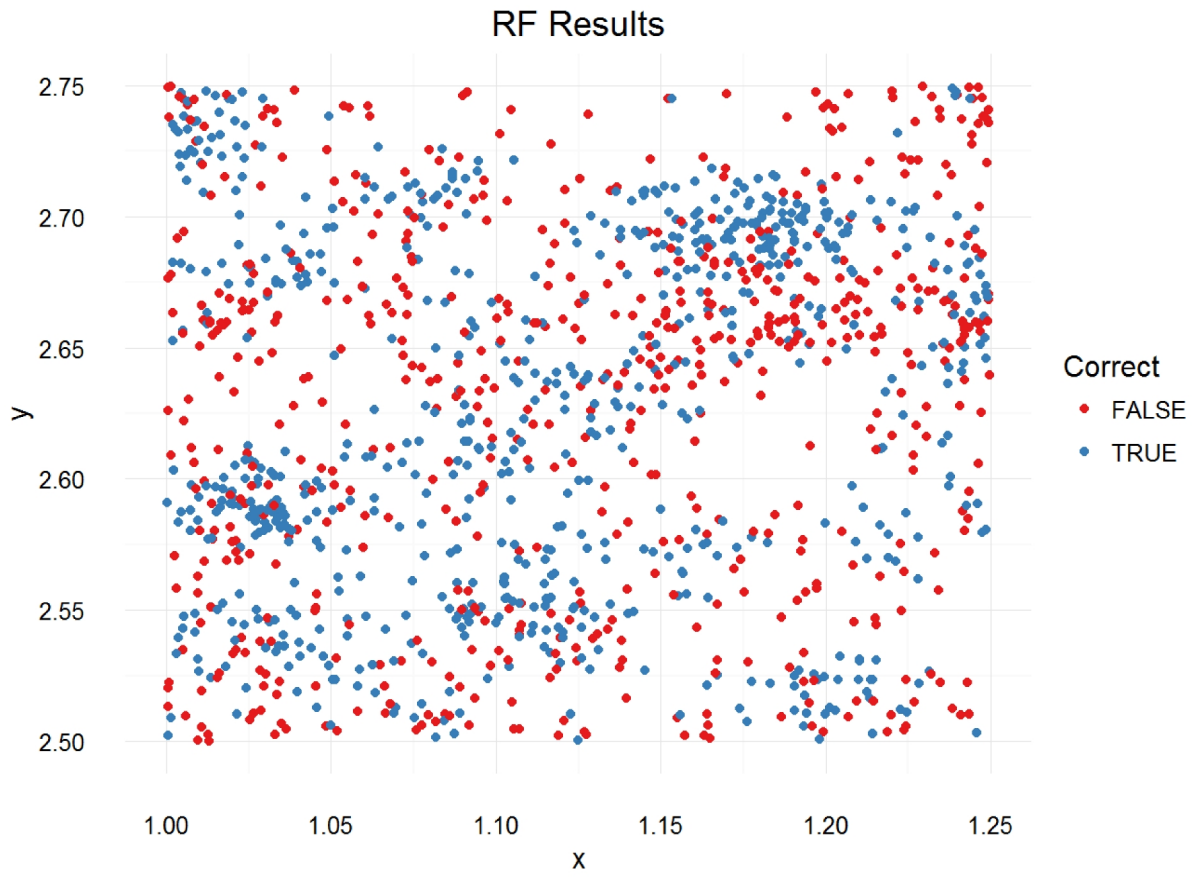
```
pred <- predict(model_rf, small_val)
pred <- pred$predictions
accuracy <- mean(pred == small_val$place_id)
accuracy
```

```
## [1] 0.5248332
```

We get an accuracy of 0.5240919.

Hey not bad! Keep in mind that the evaluation metric for this competition is mean average precision at 3 so predicting votes/probabilities by class and then counting the top three id's is guaranteed to improve our score. But for simplicity we'll just stick to accuracy. Let's take a look at the predictions on the validation set:

Kyu Cho
5/28/2016




It does seem that the correctly identified check-ins are more “clustered” while the wrongly identified ones are more uniformly distributed but other than that no clear patterns here.

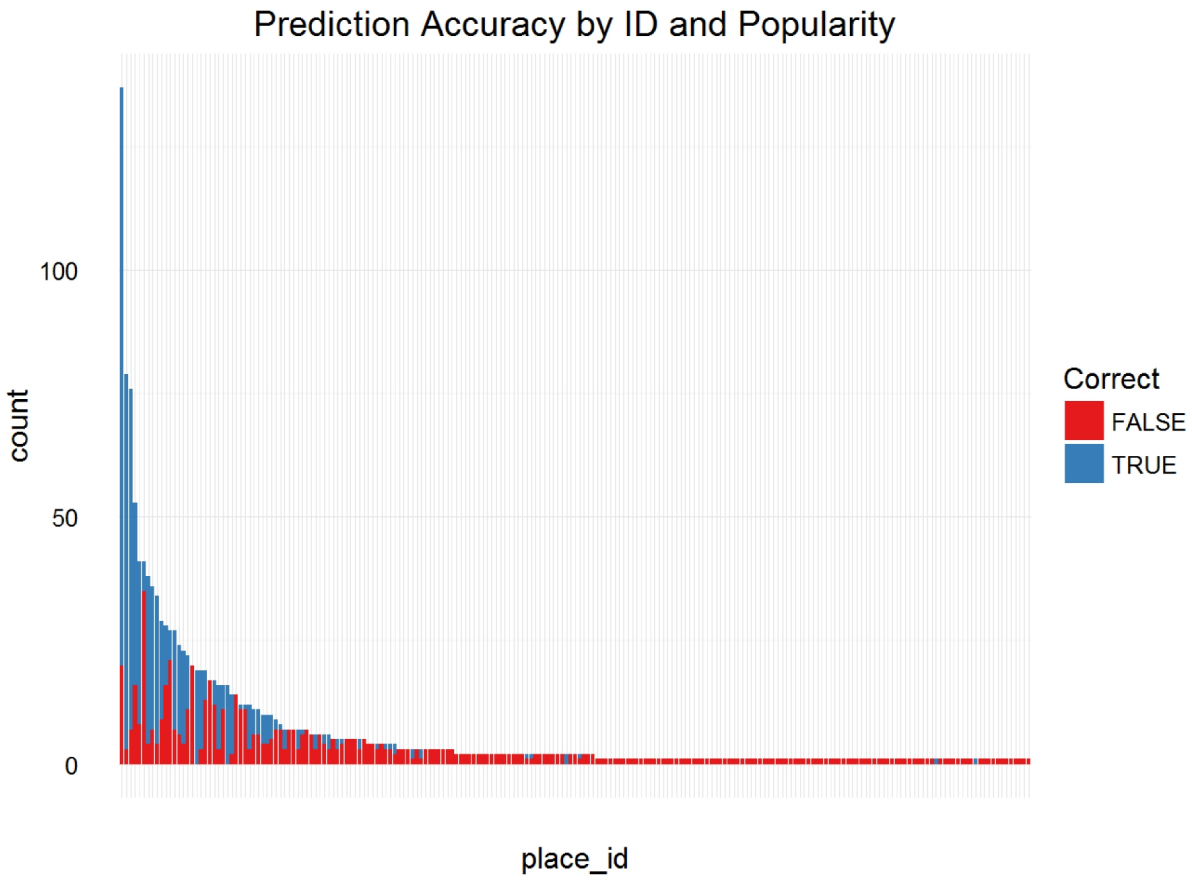
Let’s also take a look at what kind of id’s our random forest gets wrong.

To do this we will look at accuracy by id and also plot the id’s based on how often they appear in the validation set.

We see below that our model is doing actually really great on the more popular id’s(more blue on the right).

However it loses when it looks at id’s that appear only a few times.

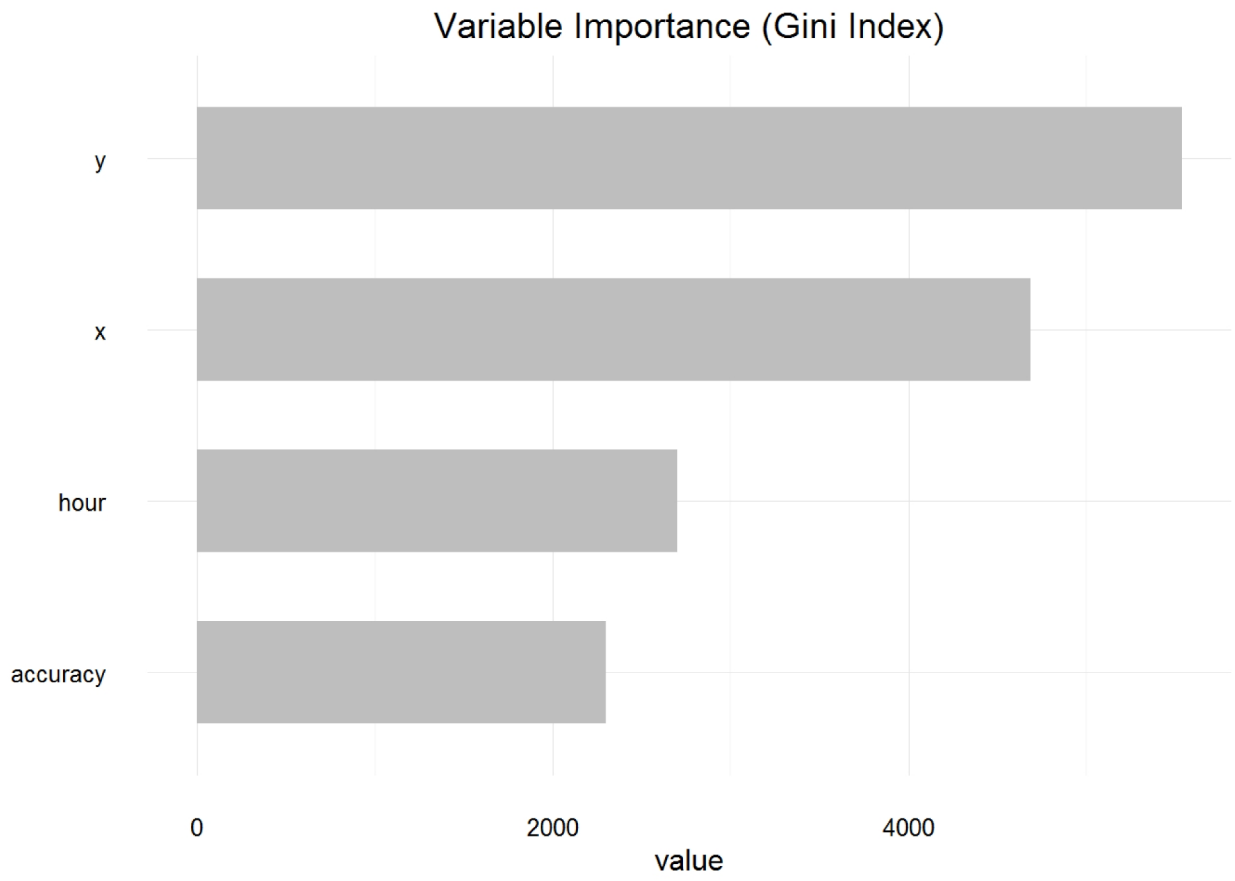
Kyu Cho
5/28/2016




Doing actually really good on the more popular id's(the blue area on the right). However it loses when it looks at id's that appear only a few times.

Let's look at the importance of our variables as well:

Kyu Cho
5/28/2016




This is quite interesting. First of all the y variable is more important than the x coordinate. This is in line with a lot of observations in other scripts: the variance by place_id is significantly higher in the x direction than in the y direction. This means that the y axis is a better predictor of place_id and the random forest figures this out on its own. Hour is also a good predictor but less so than the spatial features - this makes sense since the location of a check-in should be more important than the time of the check-in. And lastly we see that accuracy is important. Accuracy is a bit mysterious since we don't get an actual definition for it is but at least the model tells us it's somewhat important.