CS724 Course Project

Patterns of Life in NYC with Citibike

Alex Launi
01124306

July 20, 2019

Contents

1 Abstract

1.1 Objectives

1. **Does Citibike increase exploration of the city?**]
   How are New Yorkers moving around their city via citibike? Do users typically take short trips in one area, or do the bikes move to vastly different areas of the city. I will be tracking the movement of individual bikes and trying to draw conclusions about their movement.

2. **Does Citibike increase economic activity?**
   Citibike allows residents and visitors of New York City to explore areas of the city that may not be as easily reached by NYC metro alone. I will investigate whether use of Citibike corresponds to changes

in an area's economic activity. As patrons move around NYC via Citibike, do residents experience increases in taxable income?

3. **Can we make any predictions about economic growth based on Citibike data?**
   Given the previous conclusions, are we able to look at changes in Citibike patterns to predict what areas will experience growth? Does the movement of people on bikes contribute to gentrification of neighborhoods?

1.2 Description

The Citibike data is extremely large. Consider how many people are in New York City. Every ride is logged with *date, time, departing station, returning station, duration, bike id, rider sex, and rider birth year.* The station data is provided as WGS84 latitude, and longitude. The NYC open data platform provides GIS polygons providing zip code boundaries. There are 139,906 points that need to be transformed from the NAD83/Long Island, NY coordinate reference system, to WGS84. This is a computationally moderate mathematical process that could benefit from parallelization for very large numbers of coordinates. Time permitting I intend to utilize CUDA to speed up the transformation. There are two opportunities for parallelization. Clearly each point can be computed in parallel, but within each point the X and the Y can be computed independently. Initial investigation shows this process is completed in less than a second for my data. If all other goals are met, this will be parallelized, but parallelization does not appear to offer sig tnificant benefit.

Observing how bikes move around NYC is analogous to a large graph problem. There are 187 stations, and over 10,000 bicycles. I believe that the data can be viewed as a social network with stations as nodes and bicycle trips as directed edges. Processing this graph can help us draw conclusions about increasing connectivity of the city. There are over 1 million trips ***per quarter***.

1.3 Data

1.3.1 Citibike

Citibike data is compiled quarterly and available from `https://s3.amazonaws.com/tripdata/index.html`.

1.3.2 Zip code tax statistics

Tax data by zip code is compiled annually and available from `https://www.irs.gov/statistics/soi-tax-stats-individual-income-tax-statistics-2016-zip-code-data-soi`.

2 Design

2.1 Does Citibike increase exploration

The zip code system allows us to conveniently partition New York City. We will determine if cross zip code trips increase over time by taking each quarter and reviewing data month by month as follows:

```
procedure COUNT ZIP CODE CROSSINGS($G, Polygon[\,]zipcodes$)
    count = 0
    countPerZip[$zipcodes.length$]
    for $edge(v, w) \in G.E$ do
        // Result of $FindZipcodeByRayCrossing$ is cached to avoid
        // running RayCrossing algorithm for every edge.
        $v.zipcode \leftarrow FindZipcodeByRayCrossing(v.location, zipcodes)$
        $w.zipcode \leftarrow FindZipcodeByRayCrissong(w.location, zipcodes)$
        if $v.zipcode \neq w.zipcode$ then
            $count \leftarrow count + 1$
            $countPerZipcode[w.zipcode] \leftarrow countPerZipcode[w.zipcode] + 1$
        end if
```

```
13:     end for
14: end procedure
```

This algorithm will run in $O(E)$ time. The algorithm can be run in parallel, and the results combined after computation without any loss. Some slight inefficiency may occur due to having to run the Is Point In Polygon (*RayCrossing*) procedure for nodes that have been discovered on other Spark nodes, but this process is not significantly limiting and additionally offers opportunity for CUDA parallelization if it turns out to be a computational bottleneck.

The counters are implemented as a *pyspark.Accumulator* using a custom *pyspark.AccumulatorParam*, *DictParam*, that allow us to build a dictionary of counters with the following schema:

```
{
        "$zipcode": {
                "$YYYYmm": $number_of_trips
                ...
                ...
                ...
        }, ...
}
```

2.2 Does Citibike increase economic activity

Using the data we collected exploring whether or not Citibike is having an effect on inter-zip code travel we will correlate this with changes to individual income tax (per year) for each zip code. By plotting yearly total of trips coming into a zip code against individual income and plotting over time (by year) per zip code we will determine if a correlation exists between these two sets. Although we will be unable to gauge causation it should give us an indicator of coming gentrification. If people are making more bike trips into a neighborhood then we can presume that neighborhood is doing something to increase its tourism, more people are wanting to live there, home values are going up, and incomes are rising.
The coefficient of correlation (r_{xy}) can be efficiently computed in a single pass of the data as follows:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \cdot \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

This computation does not require big data since the number of zip codes is small, and they are plotted against total number of incoming trips which was calculated on the large dataset previously.

2.3 Can we make predictions about economic growth based on Citibike data

It follows from the previous section that we can attempt to use regression to make predictions about what neighborhoods will experience income growth. We will split the data into a training set and a test set. We will use the training set to compute a line of regression, and test our regression on the test set.

3 Results

3.1 Does Citibike increase exploration

A resounding yes. For every zip code in which there was enough historical data to view a trend, the number of incoming trips increased as time went on. The most significant observation was in the sharp increase, and decrease that comes with seasonal change. Usage experiences a sharp rise in the sprint through the summer, and a sharp decline after peak riding season. If I were advising Citibike about when the right time to open new stations would be, I would suggest early for construction to be available by May, to minimize idle time after initial investment.

3.2 Does Citibike increase economic activity

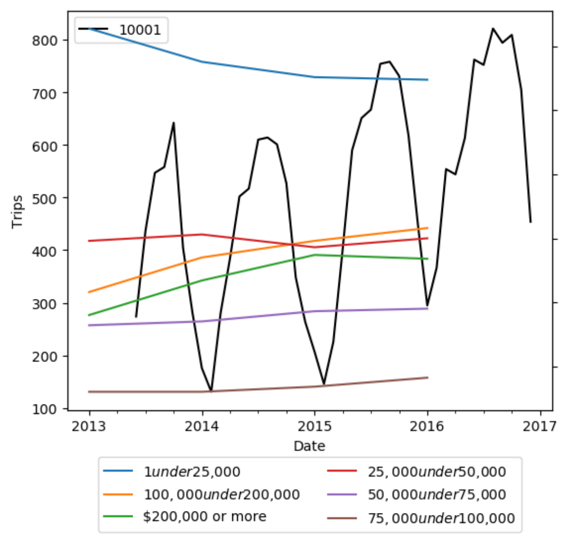
It hard to say for sure if there is a connection between Citibike usage, and economic activity in NYC. It is more likely that NYC has such a high base rate of growth that a small factor like bike sharing does not have an appreciable impact. In every zip code we examined we saw an overall increasing trend of Citibike usage. The data exhibits a peak-and-valley type distribution with high peaks in the summer time, and a quick drop off in usage as the weather gets chilly. Each year, however, more people continue to use the program even into the colder months. This trend was true for every zip code for which there was enough data to establish a trend across years.

Zipcode 10069, which is a small area on the southern end of Manhattan's Upper West side was the only zip code which experienced growth in income returns under $25,000. This growth, however was significantly outpaced by the growth in returns greater than $200,000. This change cannot, however, be attributed to Citibike. Citibike did not operate a station in this area until late 2015. This may indicate than rather than incomes trailing Citibike, Citibike trails gentrification.

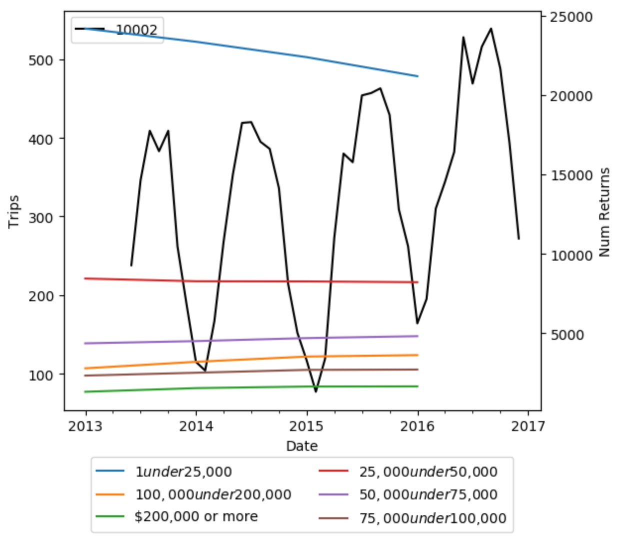
3.3 Can we make predictions about economic growth based on Citibike data

This question was not answered. The tax data did not provide enough data points to create a regression line from. Although only 2 points are needed to create a line, this line of regression would be rather meaningless. Future work may try to answer this question via some sort of aggregation of the tax data. The IRS breaks the data into income bracket categories and the number of returns within each category. This yielded 6 trend lines of 4 data points each per zip code. It is not clear how these could be aggregated into a single variable that could be plotted against number of rides. Further work could be done to answer this question.

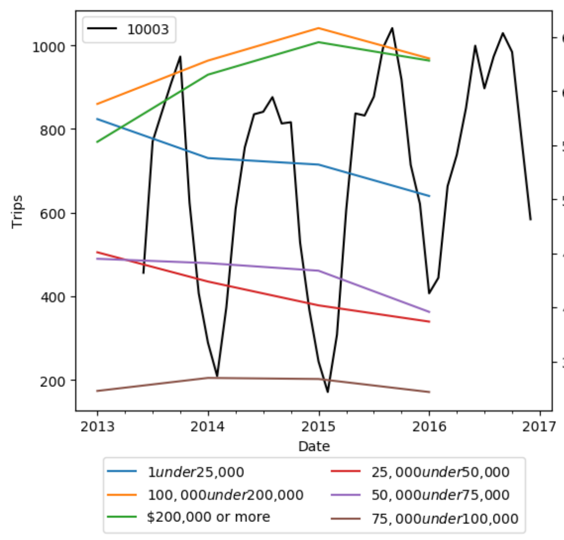
A Charts and Figures



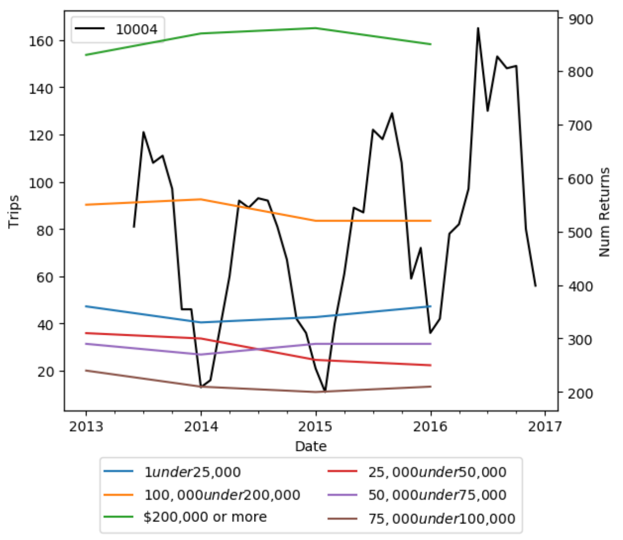
(a) 10001



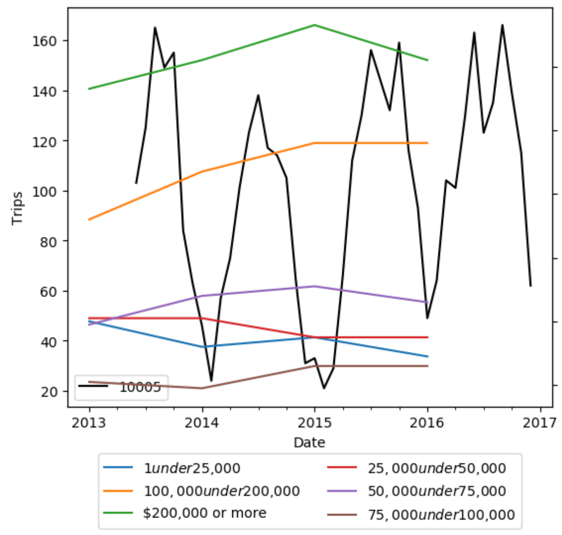
(b) 10002



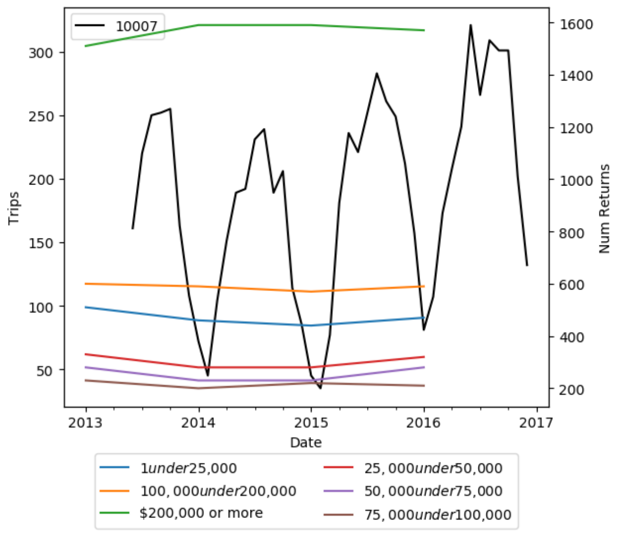
(a) 10003



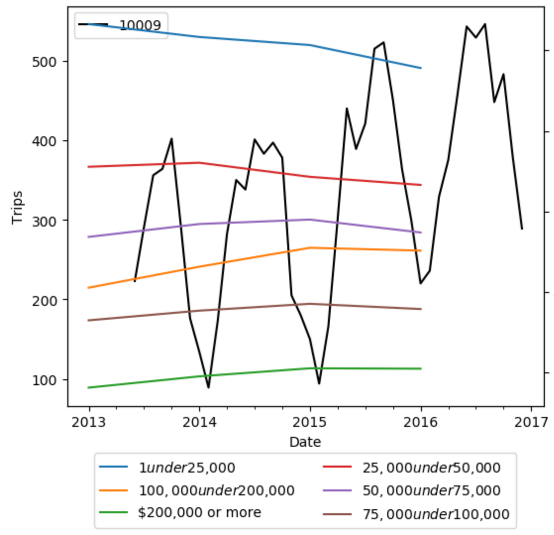
(b) 10004



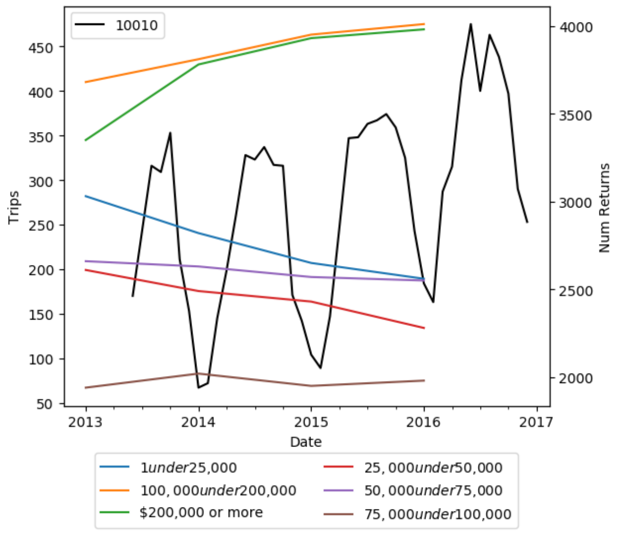
(a) 10005



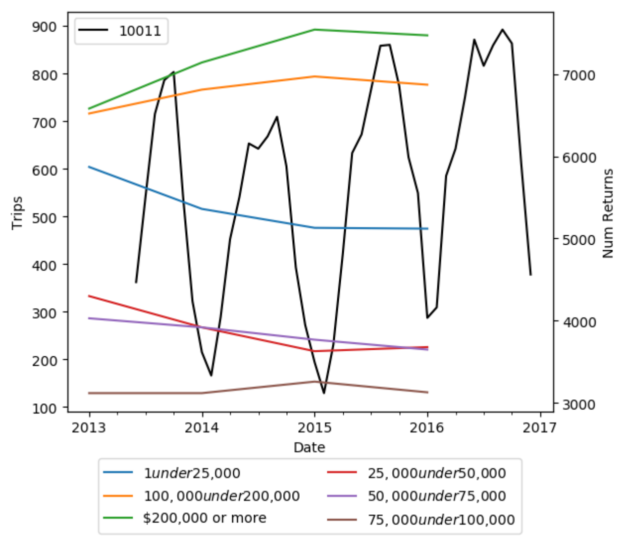
(b) 10007



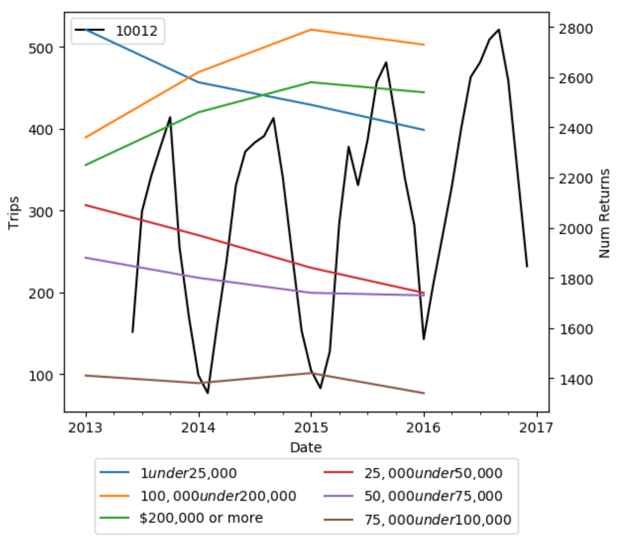
(a) 10009g



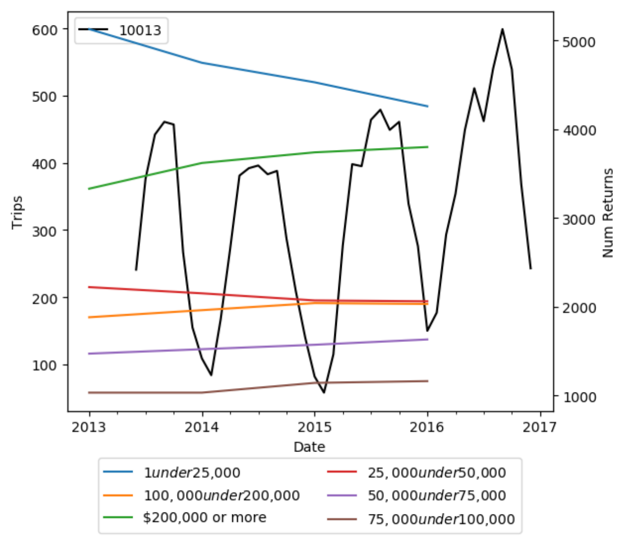
(b) 10010



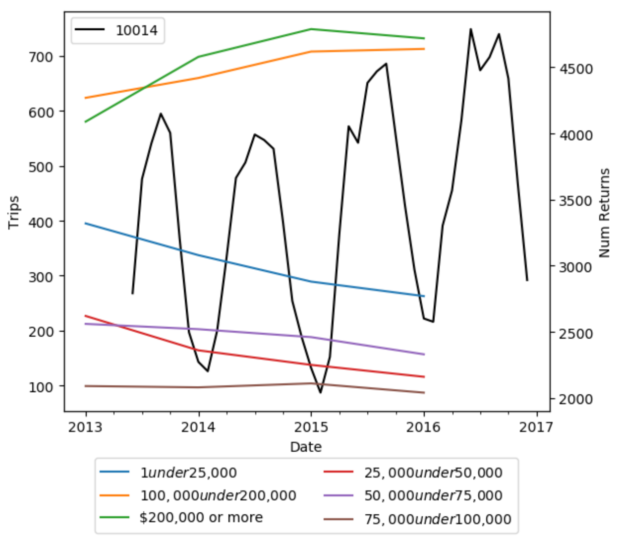
(a) 10011



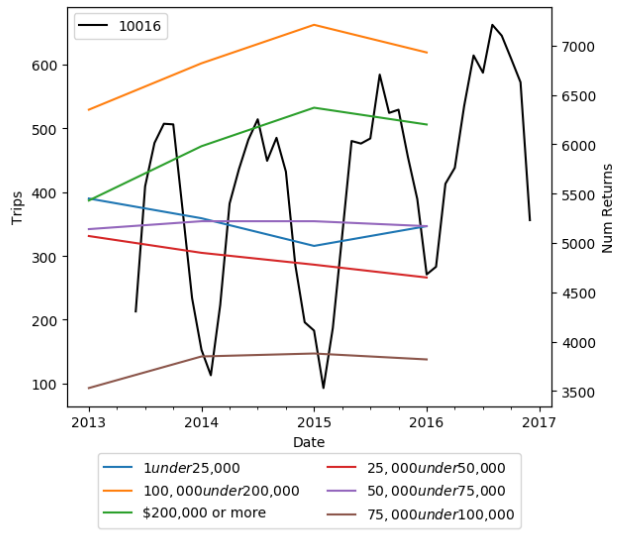
(b) 10012



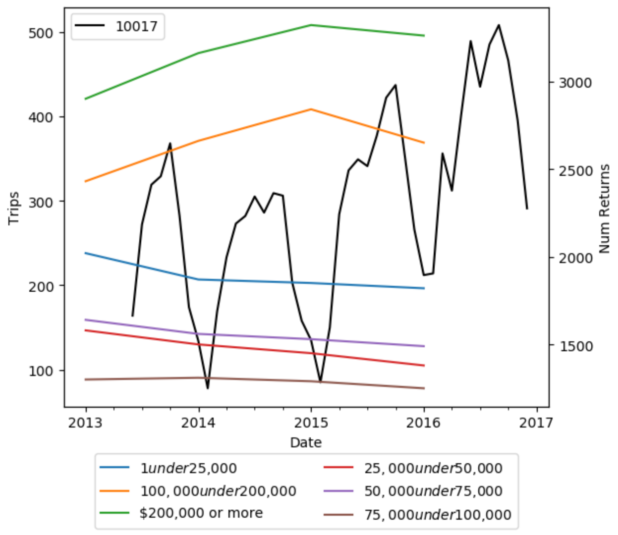
(a) 10013



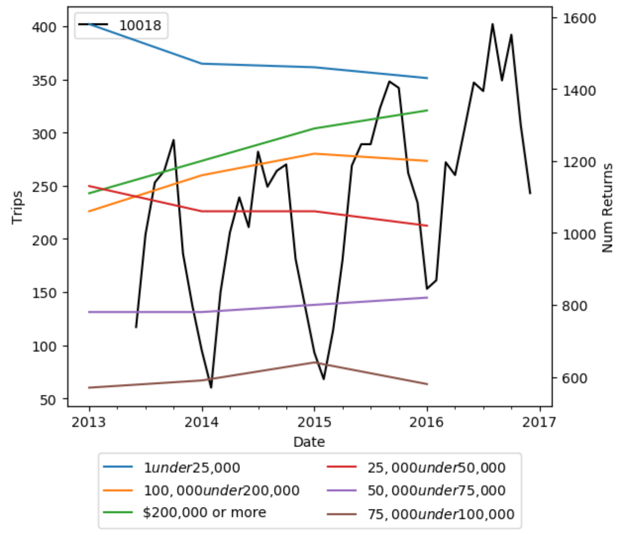
(b) 10014



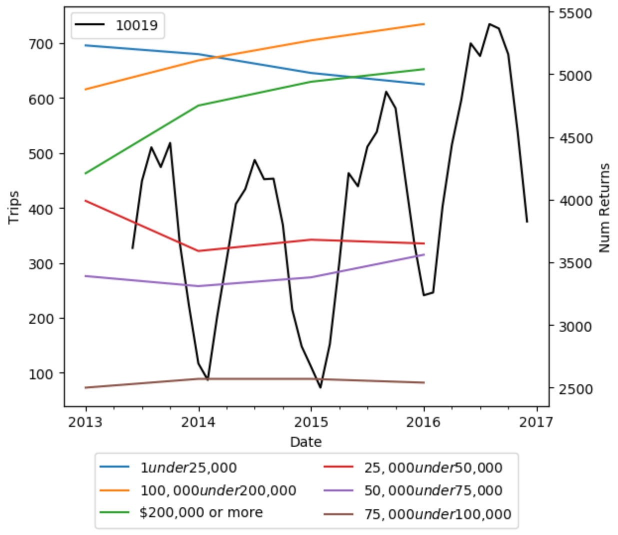
(a) 10016



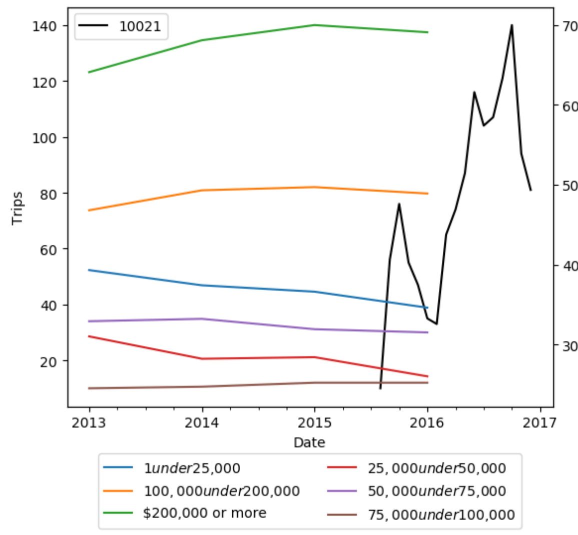
(b) 10017



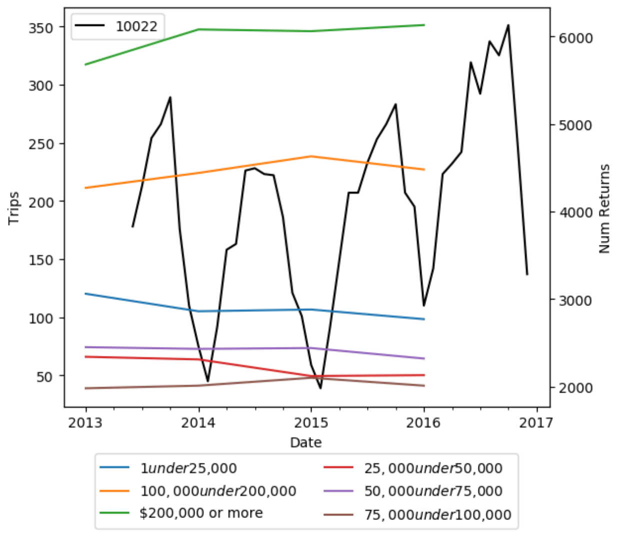
(a) 10018



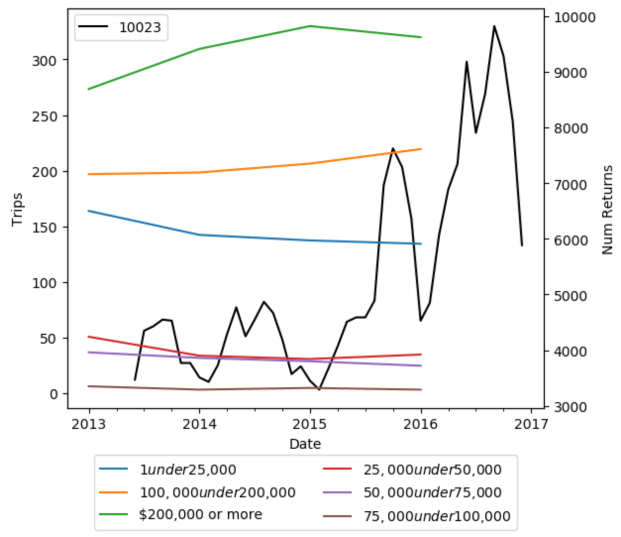
(b) 10019



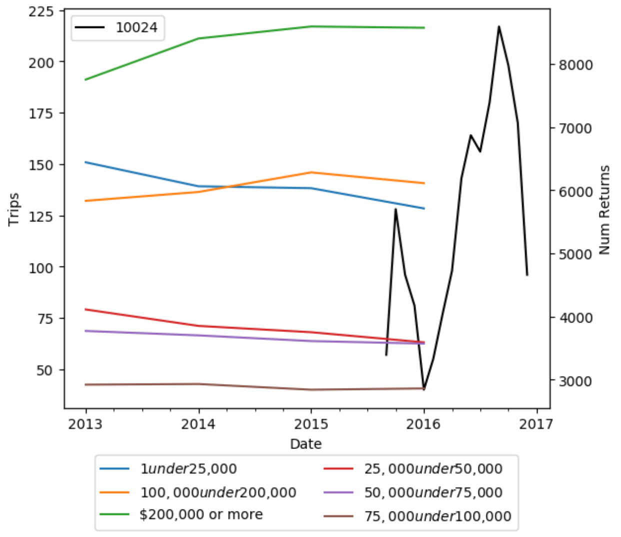
(a) 10021



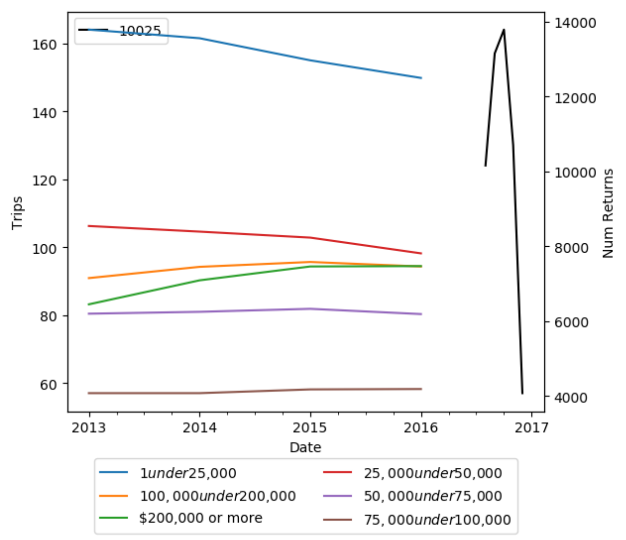
(b) 10022



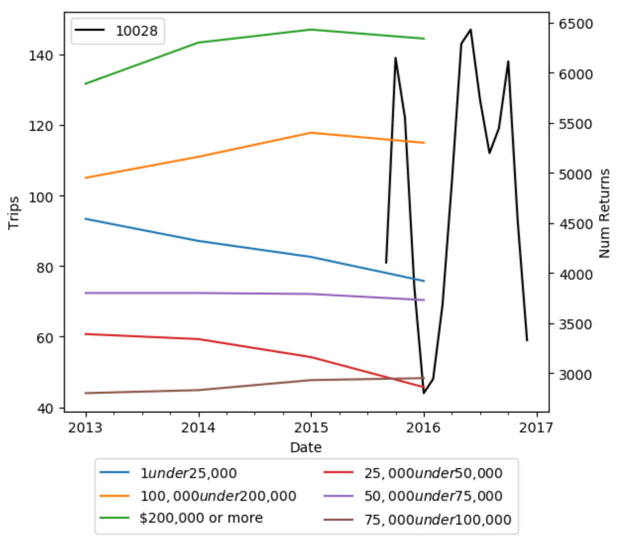
(a) 10023



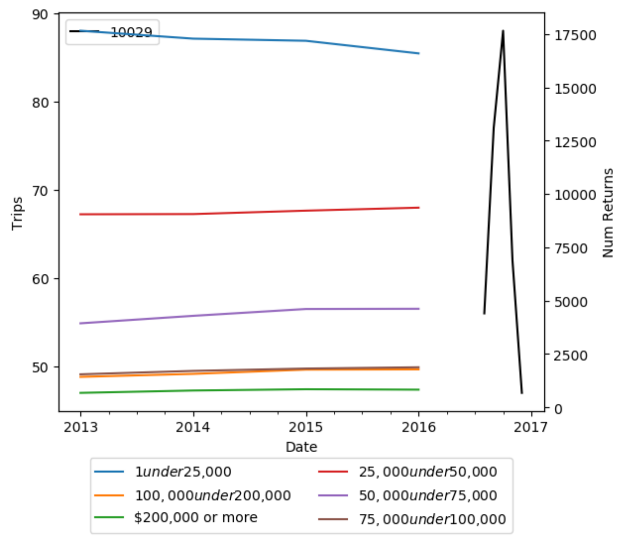
(b) 10024



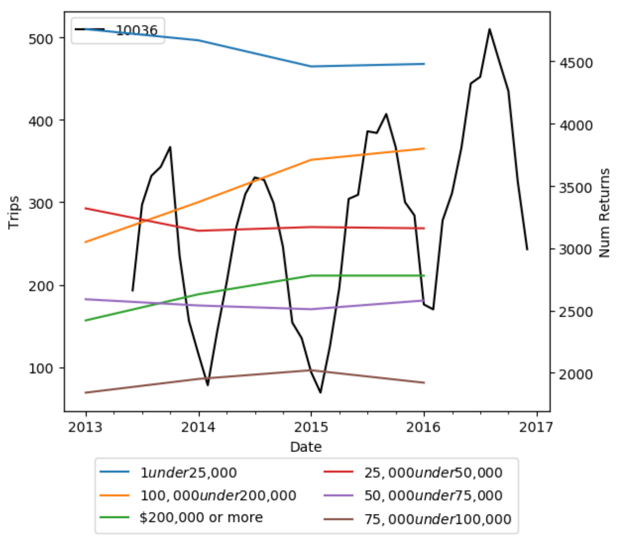
(a) 10025



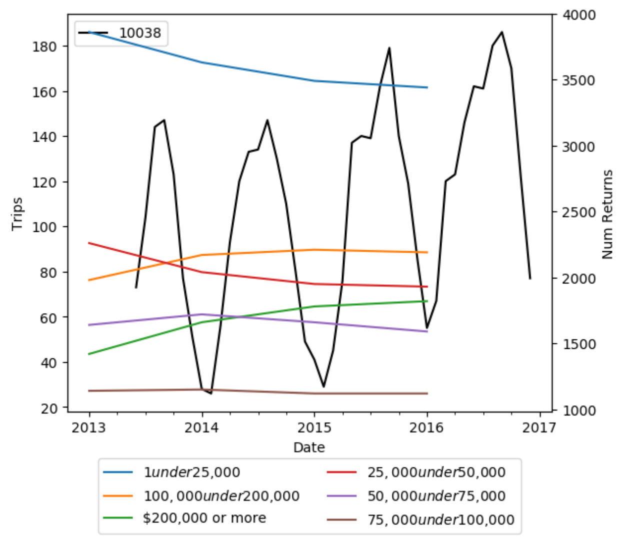
(b) 10028



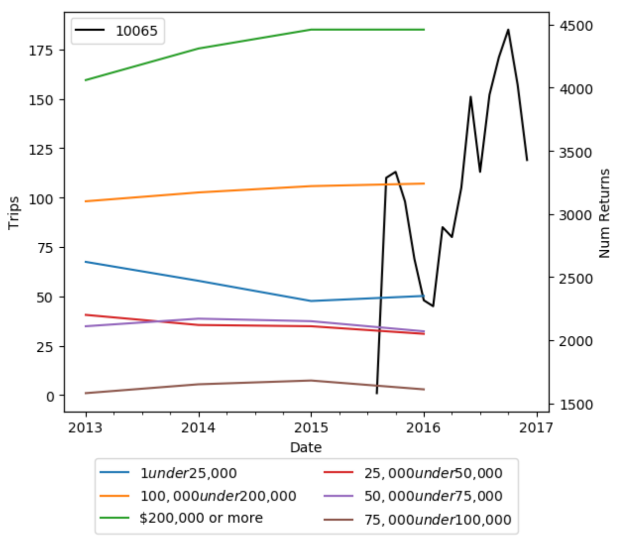
(a) 10029



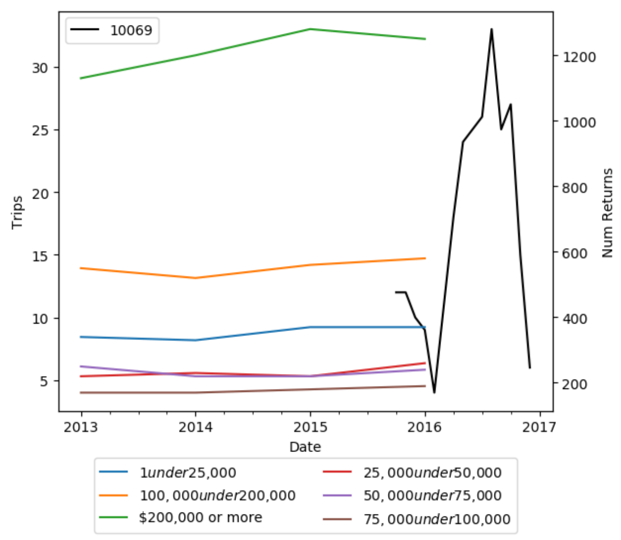
(b) 10036



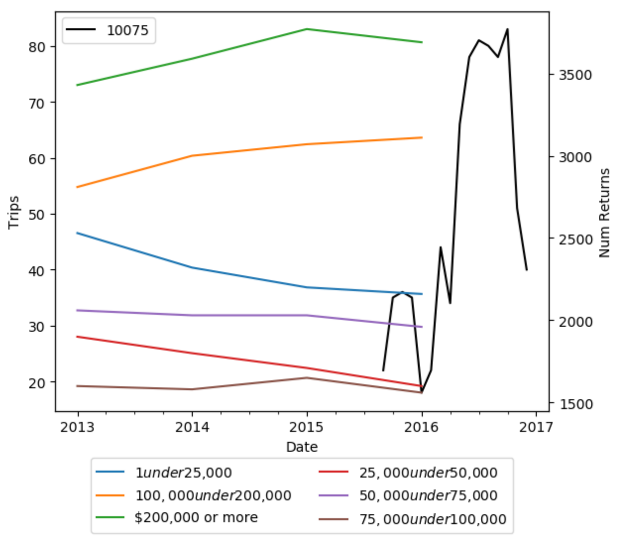
(a) 10038



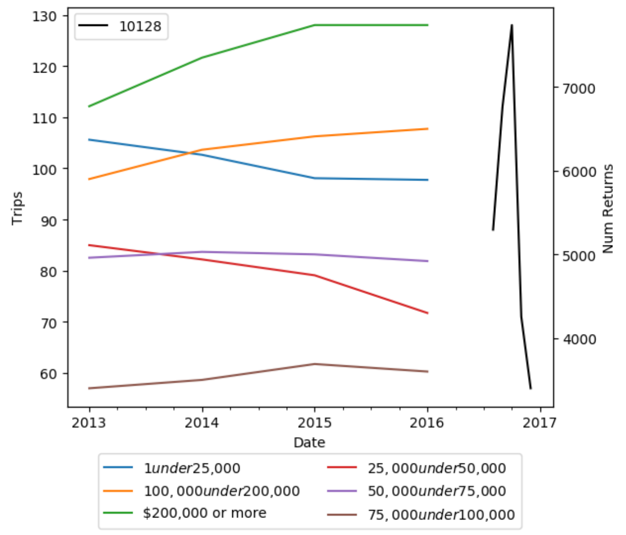
(b) 10065



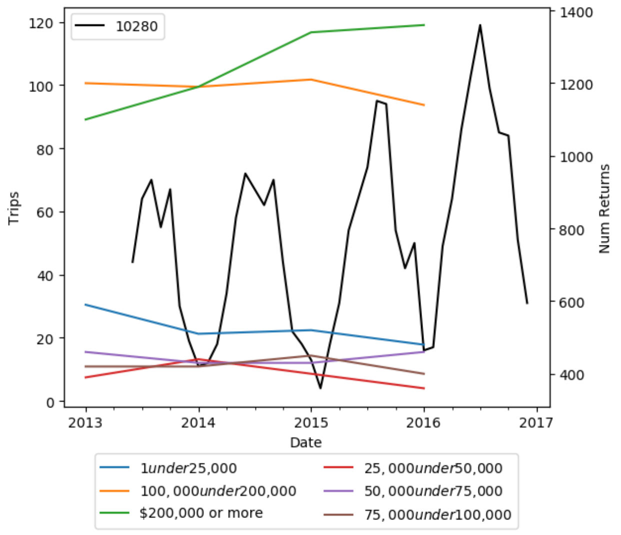
(a) 10069



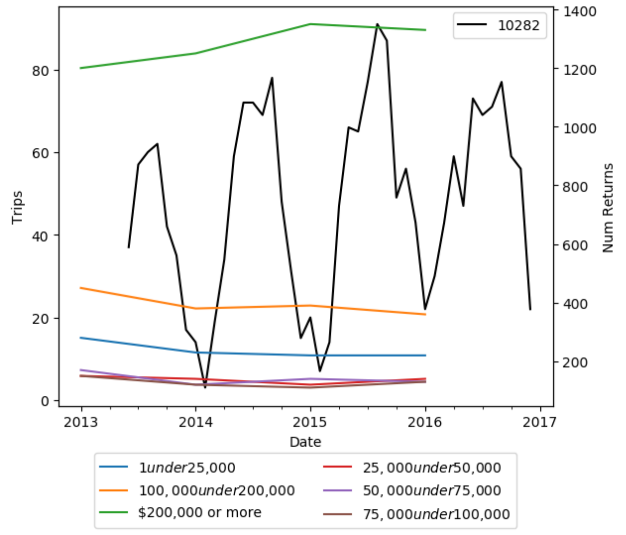
(b) 10075



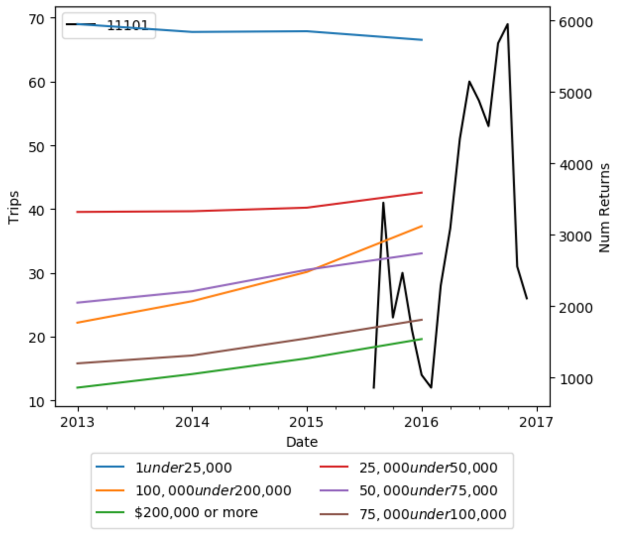
(a) 10128



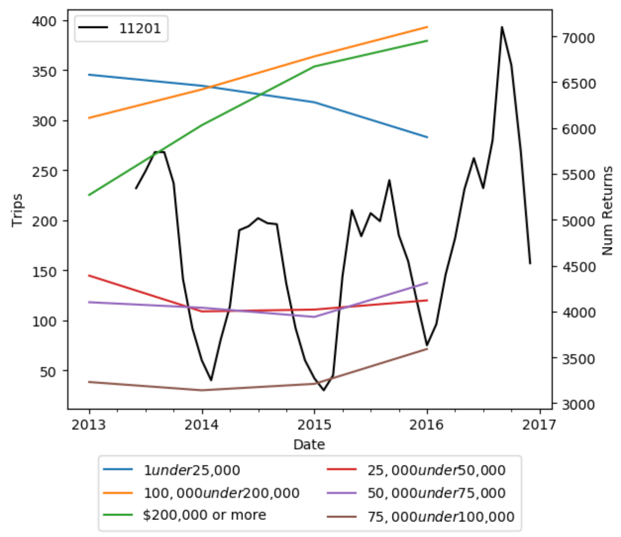
(b) 10280



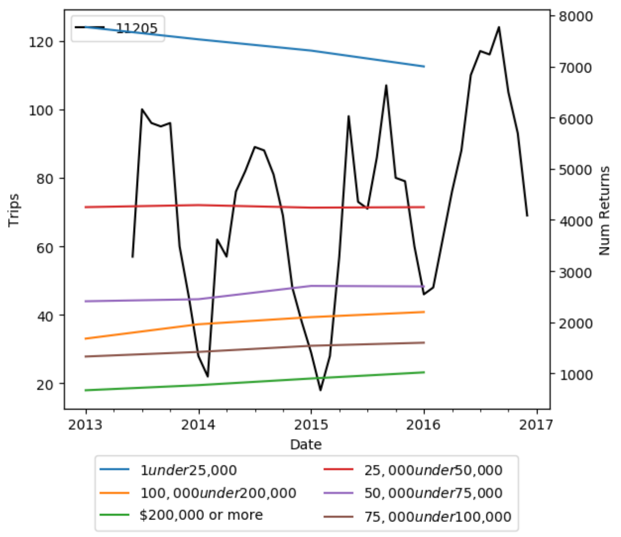
(a) 10282



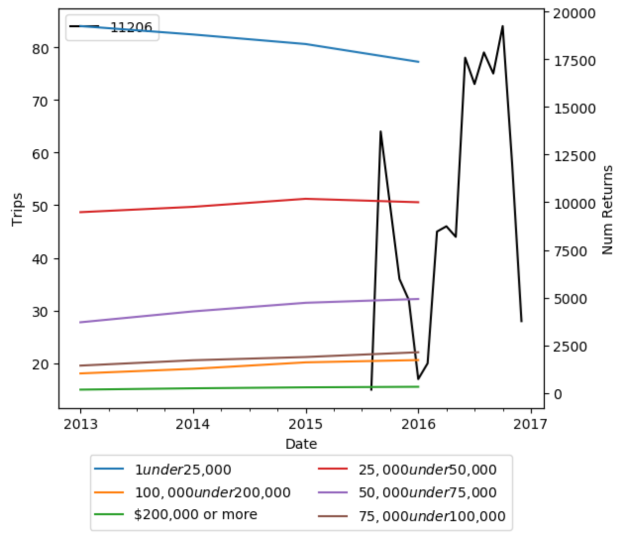
(b) 11101



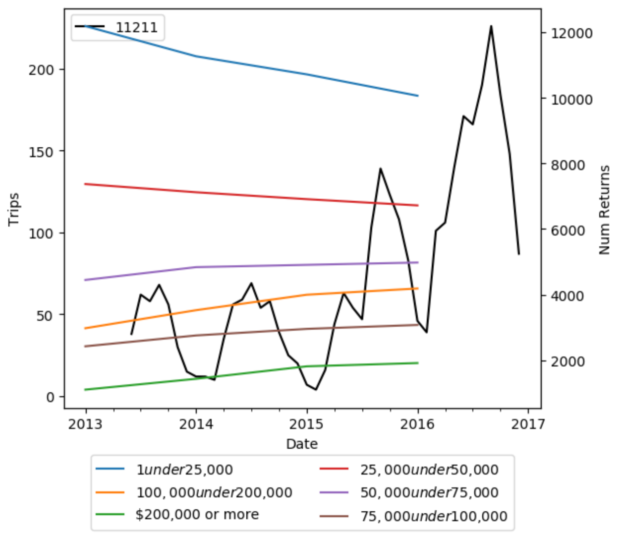
(a) 11201



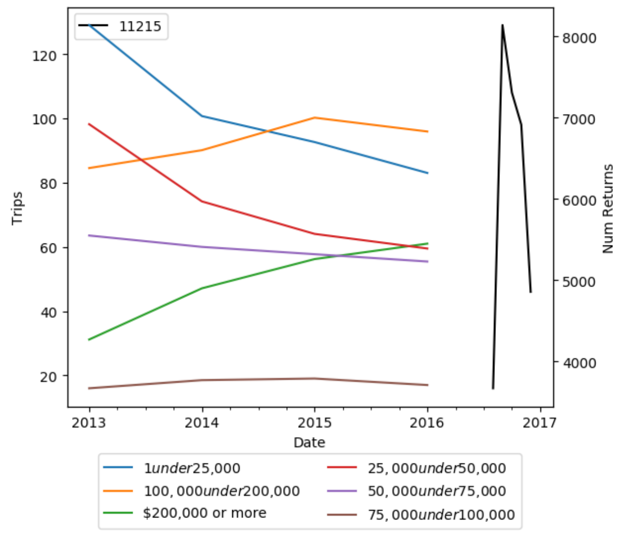
(b) 11205



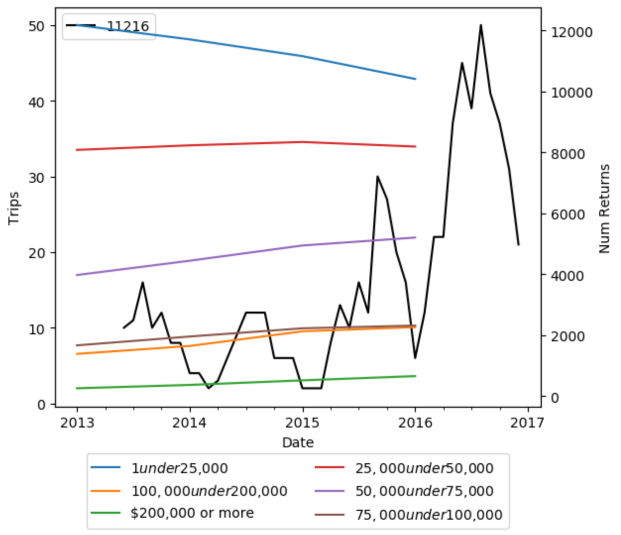
(a) 11206



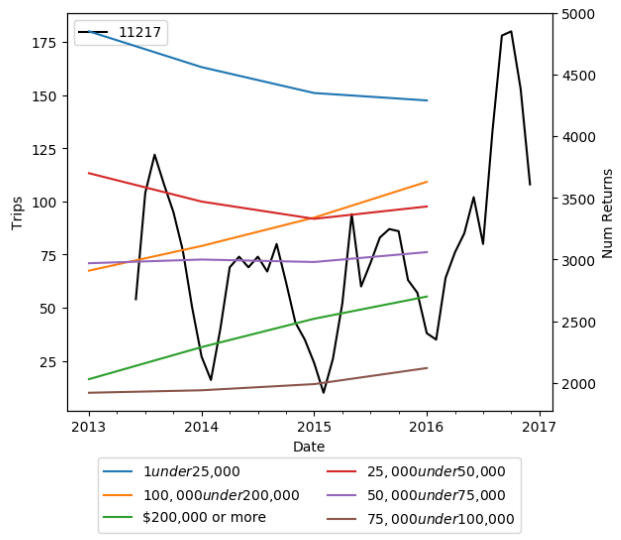
(b) 11211



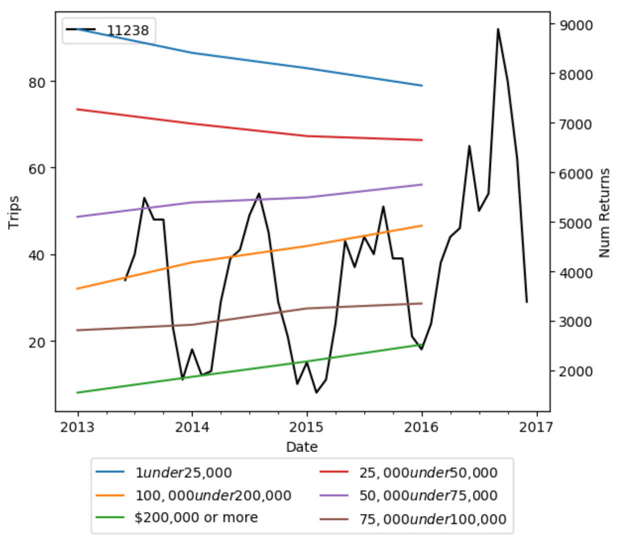
(a) 11215



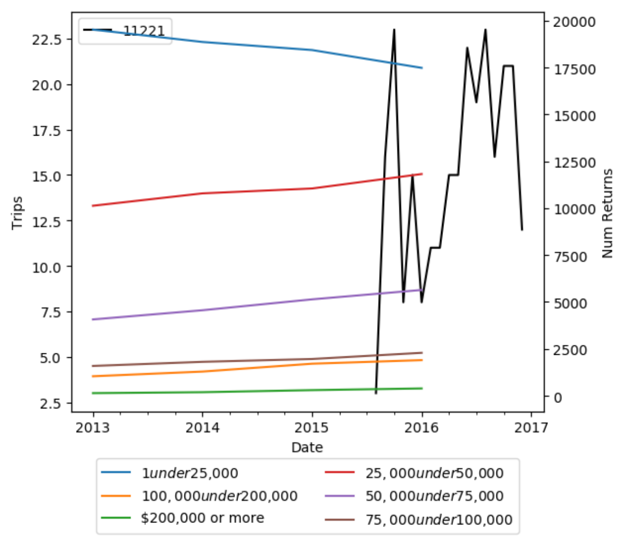
(b) 11216



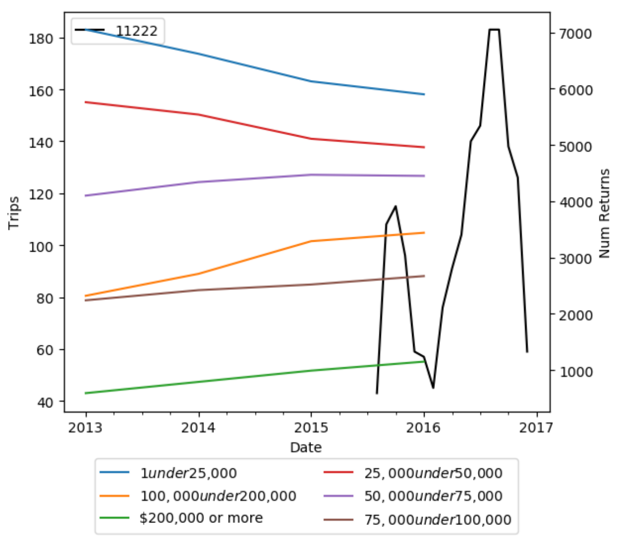
(a) 11217



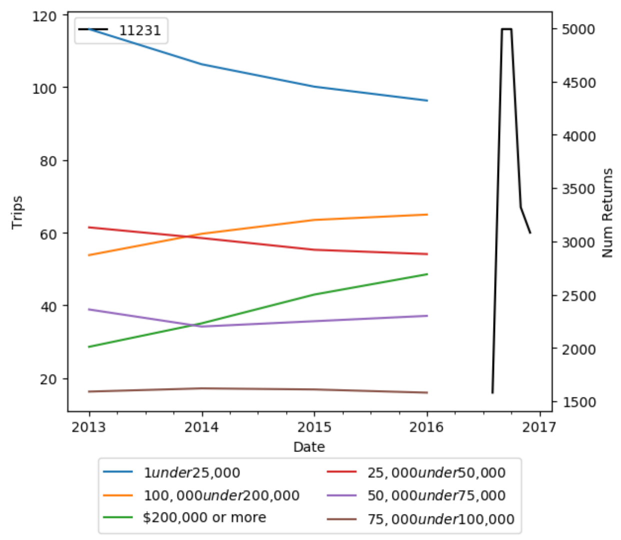
(b) 11238



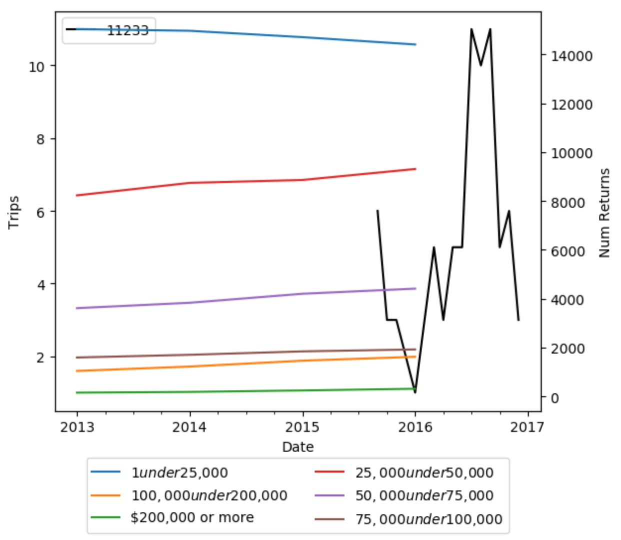
(a) 11221



(b) 11222



(a) 11231



(b) 11233

A Supplemental Algorithms

A.0.1 NAD83 to WGS84

```
/**
 * Input: Point containing NAD83/Long Island (ftUS) x, y coordinate
 * Output: Equivalent Point containing WGS84 lat/lon position.
 *
 * Adapted from PROJ.4, https://pubs.usgs.gov/pp/1395/report.pdf,
 * and http://www.epsg.org/Guidancenotes.aspx.
 **/
Point nad83_x_y_to_nad83_lat_lon(Point xy)
{
  double rho, phi, lam;

  xy.X *= XY_FACTOR_1;
  xy.Y *= XY_FACTOR_1;

  xy.X = xy.X - X0;
  xy.Y = xy.Y - Y0;

  xy.X *= RA;
  xy.Y *= RA;


  xy.Y = RHO0 - xy.Y;
  rho = hypot(xy.X, xy.Y);
  if (rho != 0.) {
    phi = phi2(pow(rho / QC, 1. / QN), PE);
    lam = atan2(xy.X, xy.Y) / QN;
  } else {
      lam = 0.;
      phi = QN > 0. ? HALFPI : -HALFPI;
  }

  xy.X = adjlon(lam + LAM0);
  xy.Y = phi;

  xy.X *= XY_FACTOR_2;
  xy.Y *= XY_FACTOR_2;

  return xy;
}

double adjlon (double lon) {
    /* Let lon slightly overshoot, to avoid spurious sign switching at the date line *
    if (fabs (lon) < PI + 1e-12)
        return lon;

    /* adjust to 0..2pi range */
    lon += PI;

    /* remove integral # of 'revolutions'*/
    lon -= TWOPI * floor(lon / TWOPI);
    /* adjust back to -pi..pi range */
    lon -= PI;

    return lon;
}

/******************************************************************************/
double phi2(double ts, double e) {
/******************************************************************************
Determine latitude angle phi-2.
Inputs:
  ts = exp(-psi) where psi is the isometric latitude (dimensionless)
  e = eccentricity of the ellipsoid (dimensionless)
Output:
  phi = geographic latitude (radians)
Here isometric latitude is defined by
  psi = log( tan(pi/4 + phi/2) *
             ( (1 - e*sin(phi)) / (1 + e*sin(phi)) )^(e/2) )
      = asinh(tan(phi)) - e * atanh(e * sin(phi))
This routine inverts this relation using the iterative scheme given
by Snyder (1987), Eqs. (7-9) - (7-11)
*******************************************************************************/
    double eccnth = .5 * e;
    double Phi = HALFPI - 2. * atan(ts);
    double con;
    int i = N_ITER;

    for(;;) {
        double dphi;
        con = e * sin(Phi);
        dphi = HALFPI - 2. * atan(ts * pow((1. - con) /
           (1. + con), eccnth)) - Phi;

        Phi += dphi;

        if (fabs(dphi) > TOL && --i)
            continue;
        break;
    }

    return Phi;
}
```

A.0.2 Point in polygon

```
bool pointInside(Point polygon[], int vertices, Point p)
{
  // must have 3 points to make a polygon
  if (vertices < 3)
  {
    cout << "less␣than␣3␣verticies" << endl;
    return false;
  }
  Point extended = Point(INF, p.Y);
  int count = 0, i = 0;

  do
  {
    int next = (i + 1) % vertices;
    if (intersect(polygon[i], polygon[next], p, extended))
    {
      if (orientation(polygon[i], p, polygon[next]) == 0) {
        return onSegment(polygon[i], p, polygon[next]);
      }

      count++;
    }
    i = next;
  } while (i != 0);

  // return true if number of intersections is odd
  // return false if number of intersections is even
  return count & 1;
}

// Takes input Points p, q, r and returns true
// if q lies on segment pr.
bool onSegment(Point p, Point q, Point r)
{
  if (q.X <= max(p.X, r.X) && q.X >= min(p.X, r.X)
        && q.Y <= max(p.Y, r.Y) && q.Y >= min(p.Y, r.Y)) {
      return true;
  }

  return false;
}

// Takes triplet p, q, r and returns
// 0 if p,q,r are colinear
// 1 if p,q,r are clockwise
// -1 0 if p,q,r are counterclockwise
int orientation(Point p, Point q, Point r)
{
  int val = (q.Y - p.Y) * (r.X - q.X) - (q.X - p.X) * (r.Y - q.Y);

  if (val == 0) return 0;

  return (val > 0) ? 1 : -1;
}

// Returns true if segments pq and rs intersect
bool intersect(Point p, Point q, Point r, Point s)
{
  int o1 = orientation(p, q, r);
  int o2 = orientation(p, q, s);
  int o3 = orientation(r, s, p);
  int o4 = orientation(r, s, q);
  // General case
  if (o1 != o2 && o3 != o4) {
    return true;
  }

  // Special cases
  // p,q,s are colinear and r lies on pq
  if (o1 == 0 && onSegment(p, r, q)) return true;

  // p,q,r are colinear and s lies on pq
  if (o2 == 0 && onSegment(p, s, q)) return true;

  // r,s,p are colinear and q lies on rs
  if (o3 == 0 && onSegment(r, p, s)) return true;

  // r,s,q are colinear and p lies on rs
  if (o4 == 0 && onSegment(r, p, q)) return true;

  // No intersection
  return false;
}
```