New York City's non-obvious nightlife hot spots

Finding non-obvious nightlife places using taxi trip data

Jens Kalshoven
Vrije Universiteit Amsterdam
De Boelelaan 1105
Amsterdam, Netherlands

Steven Raaijmakers
Vrije Universiteit Amsterdam
De Boelelaan 1105
Amsterdam, Netherlands

Eric Zonneveld
Vrije Universiteit Amsterdam
De Boelelaan 1105
Amsterdam, Netherlands

ABSTRACT

In this research we describe our process of using taxi trip data from New York City to find non-obvious nightlife hot spots in New York City. The supplied taxi data is over 50GB and thus has to be prepared first. Thereafter, we make an analysis of the prepared data which will be visualized in a web application. Using the visualization we can find non-obvious nightlife places which in turn will be analyzed using time series analysis. We choose to analyze three such hot spots with interesting background stories which are described in the experiments section.

1. INTRODUCTION

New York City's (NYC) renowned nightlife industry has a major impact on the economy of the city[3]. Over 25.000 different nightlife establishments support almost 300.000 jobs and generate a total economic output of more than 35 billion USD. An industry heavily relying on the NYC nightlife is the NYC taxi industry: their vehicles drop people off at and pick people up from multiple venues during the night.

Due to the NYC's Open Data Law [1] the history of all taxi trips arranged by the Taxi Limousine Commission (TLC) is made publicly available [2]. In our research we will preform an analysis on this data to locate popular non-obvious nightlife places in NYC. Since NYC houses over 8 milion inhabitants we expect to find hundreds of such hot spots. This information will be made feasible by creating a web based visualization of the hot spots. Using the visualization tool, the popularity of each hot spot will be correlated with different time dependent features. We choose to look for hourly, weekly, monthly and yearly trends, since those are most likely to show meaningful patterns.

2. RELATED WORK

Several studies have used the TLC trip data to find and examine emerging patterns in taxi usage. To better understand the mechanisms of urban life, such as mobility patterns, Ferreira et al.[1] create a model that allows to visualise spatial temporal taxi data. Additionally, the study demonstrates an undersupply of taxi service in Harlem while also showing that airports and major train stations are key transportation hubs. A limitation of this study, with respect to our research, is the lack of focus on nightlife. Furthermore, we decided against using their model in order to maintain a higher degree of freedom on our clustering method.

Another report about the economic impact of the NYC nightlife commissioned in 2019 by the NYC Mayor's Office of Nightlife and Entertainment[3] provides more insight into the nightlife of New York. The report divides nightlife into five sectors, those being Food Service, Bars, Arts, Venues, and Sports and Recreation.

First, the report shows us that the nightlife of NYC takes place from 18:00 to 06:00. Thereafter, for the popularity of nightlife establishments, the report finds the number of nightlife establishments in Staten Island is the lowest and is declining. Subsequently, the amount of nightlife establishments in the Bronx is also shrinking since 2015. The remaining boroughs experience a growth in nightlife establishments, with Brooklyn taking the cake and Manhattan currently possessing the most nightlife establishments.

For the taxis and Vehicles For Hire (VHF) the report demonstrates that 32% of the trips are related to the nightlife. There is a yearly growth rate of taxi and VHF pickups during 00:00 and 04:00 of 12% between 2013 and 2017. Finally, the report contains a nightlife establishment density heat map per borough, which may prove to be useful reference material for comparisons with our results.

3. RESEARCH QUESTIONS

The objective of our report is to find non-obvious places leading to the research question: "What are non-obvious places of nightlife in New York?"

To answer this question we make use of two programming languages: Java and Python3. With Java we make use of the Apache Spark framework in order to perform multi-threaded operations on the dataset as described in section 5.1 and 5.2.1. Python3 is used when Apache Spark shows to not perform well, such as when dealing with iterations or when it is unable to cope with a large amount of data, described in further detail in section 5.2.2 and 5.2.3. At last, the data will be represented using a GeoJSON format and is visualized using Mapbox GL JS, as explained in section 5.3. The solution we created is able to handle more data than handled in this project, but there would be an unknown increase in time depending on the increase in data.

Since the answer to our research question will be ambiguous we will visualize our results to help giving an answer to the question. Besides, the research question has to be specified since we are working with a limited amount of resources, described in section 4.

[1]NYC Open Data: `https://opendata.cityofnewyork.us/open-data-law/`
[2]TLC trip record data: `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

4. BACKGROUND

Our research is dependent on two dataset. The taxi dataset contains all taxi trips provided by the TLC and includes various types of information about each taxi trip such as vendor id, passenger count, payment type and fare amount. For our research we are only interested in a subset of these features, namely the trip's pickup location, drop off location and the corresponding DateTime objects.

In this dataset the format of the location feature has changed over time. Initially, the locations are denoted by geographical coordinates, but after June 2016 the geographical coordinates are replaced by IDs. These IDs represent 263 unique taxi zones in NYC which vary in size, shown in Figure 1. With respect to our research, the taxi zones are too large to make meaningful statements about. We therefore choose to only consider taxi trips dating from 2009 to and including 2015.

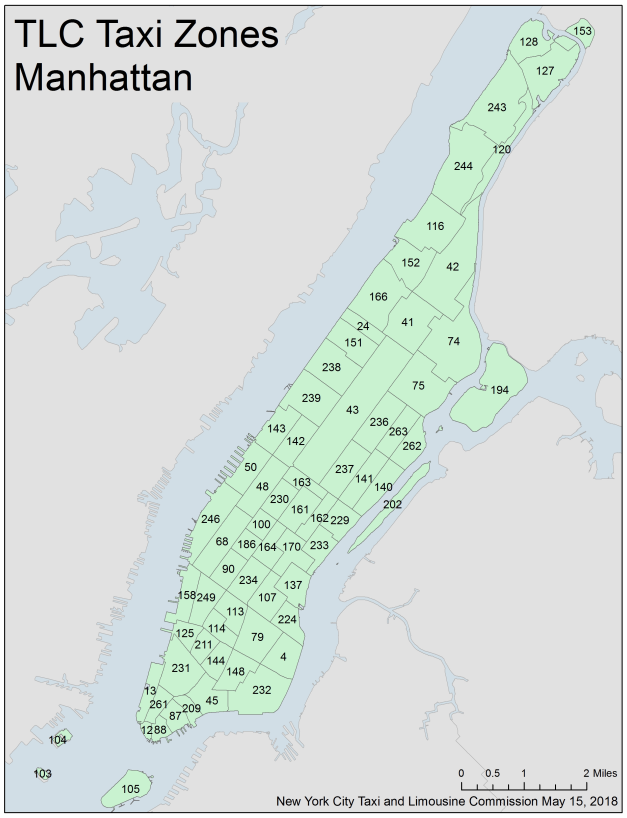


Figure 1: TLC Taxi Zones for Manhattan (footnote 2).

The second dataset contains information about bars throughout NYC and is obtained via the Kaggle competition by E. Vasilev, "2016 Parties in New York" [3]. This dataset contains ZIP codes, boroughs and coordinates of 2441 bars in NYC, which roughly responds to the number of bars in NYC found the literature [3].

Additionally, the Kaggle data does not contain all NYC bars but rather bars, clubs or restaurants that have had at least ten noise complaints between 2011 and 2016. We assume that these complaints are mostly focused on bars and clubs, since restaurants are less likely to play loud music during the night. Therefore, using this dataset we focus on the bars and clubs part of the nightlife in NYC.

[3]`https://www.kaggle.com/somesnm/partynyc`

5. PROJECT SETUP

The supplied taxi data is over 50 GB, hence we divided the project into different phases: data preparation, data analysis and visualization of the analysis. In Figure 2 an overview of the processes can be found with the different colors denoting the different phases.

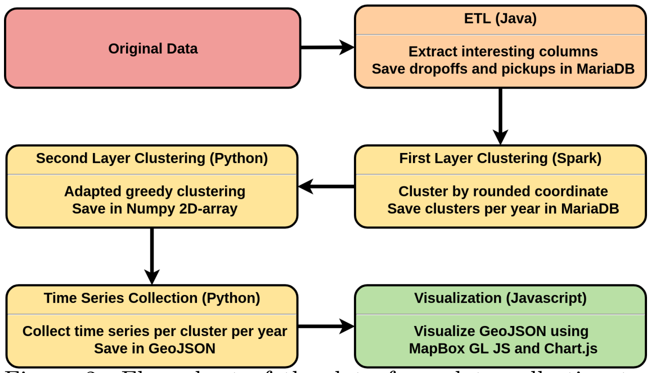

Original Data
ETL (Java)
Extract interesting columns
Save dropoffs and pickups in MariaDB
First Layer Clustering (Spark)
Cluster by rounded coordinate
Save clusters per year in MariaDB
Second Layer Clustering (Python)
Adapted greedy clustering
Save in Numpy 2D-array
Time Series Collection (Python)
Collect time series per cluster per year
Save in GeoJSON
Visualization (Javascript)
Visualize GeoJSON using
MapBox GL JS and Chart.js

Figure 2: Flow chart of the data from data collection to visualization.

Both the data preparation and analysis will be executed on one of our own PCs with the following specifications:

- Intel Core i7-7700K 4.2 GHz Quad-Core Processor,
- Corsair Vengeance LPX 32 GB (4 x 8 GB) DDR4-3000 Memory,
- Samsung 860 Evo 2 TB 2.5" Solid State Drive.

The visualization will be a web based application which runs client side, build for Google Chrome.

5.1 Data preparation

For the data preparation we start with six threads with each their own input stream of taxi trips (footnote 2). Each input stream contains monthly trip data since the data supplied by the TLC is separated by month. Considering we are using six threads we can process six months concurrently.

Using over six threads will decrease the performance on the PC, as each thread gets delivered 2 MB/s. The bandwidth is limited to 100 Mb/s, thus six threads will be the maximum before the bandwidth will be exceeded.

In every thread the features that are relevant to our research will be extracted from the taxi data, which are:

- Pickup DateTime,
- Pickup location,
- drop off DateTime,
- drop off location,
- Trip distance.

Using these features we can remove invalid trips according to the following requirements:

1. The difference between the pickup and drop off time should be at least 30 seconds.

2. The trip distance should be a value between 0.1 and 500 miles.

3. The pickup location should be a coordinate within NYC.

4. The drop off location should be a coordinate within NYC.

On a sample of 1 million trips these requirements lead to 5% invalid trips.

The remaining valid trips will be written into CSV files, two per month. One CSV contains features concerning the pickup (time and location) while the other CSV contains features concerning the drop off. Hereby it is important to note that requirement 3 and 4 are evaluated separately, e.g. when requirement 3 is met while 4 is not, the drop off data will not be written into the drop off CSV but the pickup data will be written to the pickup CSV.

We choose to separate trips into two tables since we are interested in pickups and drop offs at different time ranges, explained in section 5.2.1. This will simplify the queries and decrease query time. The CSVs will be put into MariaDB and since this uses the limited resources of MariaDB it will take a significant amount of time querying on larger databases. Saving the tables separately allows to load each table independently and perform queries on using a multi-threaded function. An analysis performed on the complete dataset will overload the RAM whilst using separated tables will increase the performance.

The DateTime features will be separated into four features, denoting year, month, day of week and hour respectively. Hereby, the DateTime information loses precision considering the minutes are discarded and the day number per month will be mapped to a day of the week. For our project this will not be an issue since our research focuses on hourly, weekly, monthly and yearly trends. Furthermore, it will benefit our performance in multiple ways: a conversion for the DateTime feature becomes superfluous and the new features will only take up five bytes in total instead of eight.

Finally, we group CSVs per year per trip type (drop off or pickup) and write them to tables in MariaDB. Specifically, one table will represent one year and one trip type. In Figure 3 the structure of these tables is shown. Each MariaDB table contains about 110 million taxi trips taking up to about 4.2GB. This is a significant decrease from the 50GB of data per year in the original dataset.

The writing to CSVs is used as an intermediate step because they will serve as a backup of the collected trips. For instance, if for any reason the connection to the MariaDB fails we can later add it manually using the CSV files.

The complete data preparation is performed using Java over Apache Spark since it has better performance for stream processing. Furthermore, a Java function can be called by a Spark function, so a single Spark function can call all steps instead of the user calling all functions manually.

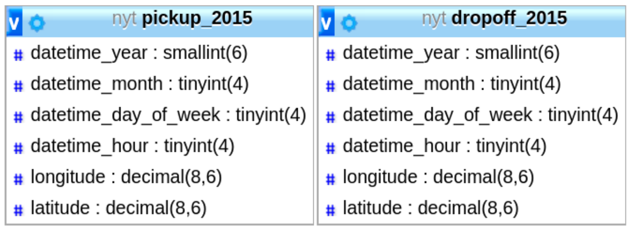
5.2 Data analysis

The data analysis is divided into three parts:

1. First layer clustering

2. Second layer clustering

3. Time series collection



nyt **pickup_2015**	nyt **dropoff_2015**
datetime_year : smallint(6)	datetime_year : smallint(6)
datetime_month : tinyint(4)	datetime_month : tinyint(4)
datetime_day_of_week : tinyint(4)	datetime_day_of_week : tinyint(4)
datetime_hour : tinyint(4)	datetime_hour : tinyint(4)
longitude : decimal(8,6)	longitude : decimal(8,6)
latitude : decimal(8,6)	latitude : decimal(8,6)

Figure 3: Database structure of one year in MariaDB.

Each part is written in a different programming language making optimal use of their different features, as described per section.

5.2.1 *First layer clustering*

The first layer of clustering is achieved using Apache Spark [4]. In this process all trip coordinates will be rounded off to four decimals. We choose to work with four decimals since in NYC a 0.0001 difference in latitude equals roughly 11.1 meters, while a 0.0001 difference in longitude equals about 8.24 meters. Rounding off the decimals will therefore map points to clusters of 11 by 8 meter. This size is suitable since avenues and major cross streets are typically 30 meters wide and standard cross streets are typically 18 meters [5]. The rounded coordinates will therefore still contain proportionate precision to determine on which street side the trip is located. In Figure 4 an example of the rounding off process is illustrated. It shows a trip ending at 40.76296, -73.97414 being mapped to the cluster 40.7630, -73.9741 on the crossing of 5th Avenue and 57th Street.

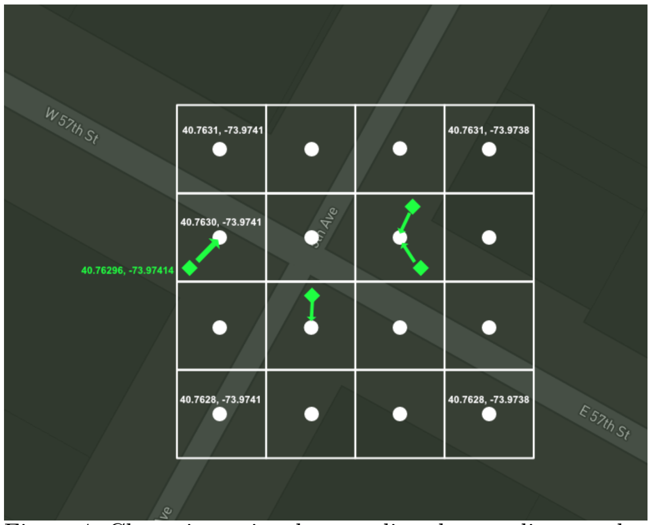


Figure 4: Clustering points by rounding the coordinates: the green coordinates are rounded to 4 decimals and clustered at the nearest point.

As shown in the figure, multiple points will be mapped to the same coordinates to form clusters. We choose to group the clusters per year, resulting in the candidate hot spots. For every cluster i we count the number of day trips N^i_{day} and night trips N^i_{night}. A trip is classified as either when it

[4] `https://spark.apache.org/`

[5] Making the plan: `http://thegreatestgrid.mcny.org/greatest-grid/making-the-plan/12`

falls into certain time ranges which we base on the following assumptions:

1. A survey of the U.S. Department of Agriculture from 1998 showing that most Americans finished dinner after 22:00 [2], shown in Figure 5.

2. Bars in NYC are not allowed to serve alcohol past 04:00 REF.

Since we only focus on bars and clubs, we cannot use the nightlife definition of the literature. The literature encompasses more than just bars and clubs for determining the times that nightlife starts and ends. Therefore, we use the above assumptions to postulate that the NYC nightlife will start after most diners are finished, which is around 22:00, and ends after the bars close, which is probably shortly after 04:00. We also postulate that people will arrive at nightlife locations at least one hour prior to closing time and will leave at latest two hours after the proposed closing time. If the hour feature of a trip meets these requirements it is classified as a night trip and a day trip otherwise. The specific time ranges are illustrated in Table 1

		drop off	**Pickup**
Day	start	03:00	05:00
	end	21:59	23:59
Night	start	22:00	00:00
	end	02:59	05:59

Table 1: Time range for day and night

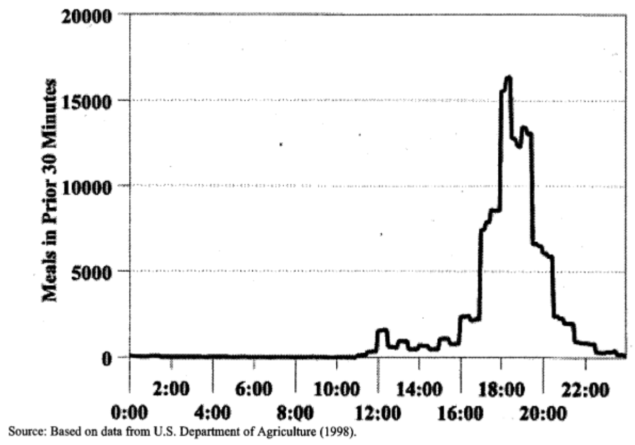


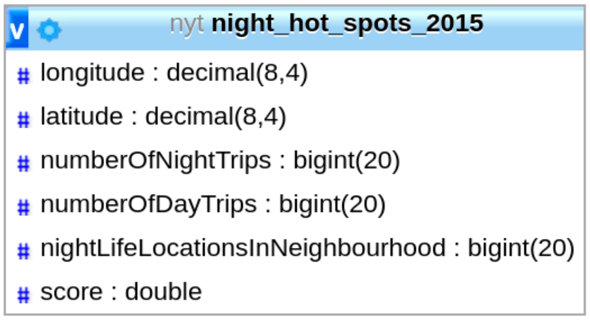
Figure 5: Distribution of the dinner times in the U.S., according to survey data by the U.S. Department of Agriculture (1998).

After making the distinction between night and day trips we set a minimum cluster size of 100 concerning the arriving or departing night trips, denoted by $N^i_{night} \geq 100$. This will filter out most clusters caused by commuters. However, some of such clusters will still get through but will most likely be removed by a requirement set during the second layer clustering, as explained in section 5.2.2.

Another requirement for a cluster i is $N^i_{night} \geq N^i_{day}$ whereas we are explicitly looking for night hot spots. Clusters that are popular during both day and night are filtered out this way, just as taxi heavy locations like airports. This requirement leaves us with locations of which we can assure that people are more likely to visit them during the night versus during the day, which we classify as being a nightlife hot spot.

Finally, we assign the property N^i_{bars} denoting the amount of bars or clubs nearby. Using the bars dataset we calculate the Euclidean distance between each bar and cluster and join them if the distance < 0.0005, equaling an oval radius of 55 by 45 meters. The reason behind this is taxis are not always able to stop directly in front of the intended destination. Using this method will give taxi drivers a radius of 55 by 45 meters to park around the location. Increasing the distance between a bar and trip would not be desirable since city blocks are typically 61 meters wide (footnote 5) thus creating the possibility of linking trips to bars on the other side of city blocks.

All clusters and corresponding properties will be written into a new MariaDB table with a structure illustrated in Figure 6. This table contains about 4.000 clusters, which translates to a size of 300KB per year. We save these clusters per year because this allows us to visualize the changing locations of the clusters over the years later.


nyt **night_hot_spots_2015**

- longitude : decimal(8,4)
- latitude : decimal(8,4)
- numberOfNightTrips : bigint(20)
- numberOfDayTrips : bigint(20)
- nightLifeLocationsInNeighbourhood : bigint(20)
- score : double

Figure 6: Structure of the night hot spots table in MariaDB

We choose to use Spark since it is able to multi-thread most of the operations performed on dataframes. This increases the performance significantly when compared to a sequential version.

5.2.2 Second layer clustering

The second layer clustering is accomplished using Python3 [6] and the NumPy library [7]. The second layer clustering is necessary for the visualization since the 11 by 8 meter clusters will still lead to multiple points around one bar, as shown in Figure 7.

The second layer clustering is accomplished using an adapted form of a greedy clustering algorithm, which works as follows:

1. Take the first cluster in the list of clusters that you have,

2. Find all the clusters with a Euclidean distance of less than 0.0005 (a 45x55 oval radius),

3. Find the cluster with the highest number of night trips within this set of clusters,

[6]`https://www.python.org/download/releases/3.0/`
[7]`https://numpy.org/`

4. (a) If the chosen cluster is the one with the highest number of night trips use this cluster as the middle, remove all other clusters from the list and go back to step 1 as long as there are clusters left,

   (b) Else, take the cluster with the highest number of night trips as the new "chosen" cluster and repeat from step 2.

During this process another requirement is set: a second layer cluster should contain at least three first layer clusters. This is necessary, since we notice some first layer clusters are still located at odd positions. These are most likely flats where lots of people live, while bars often have multiple smaller clusters around them.

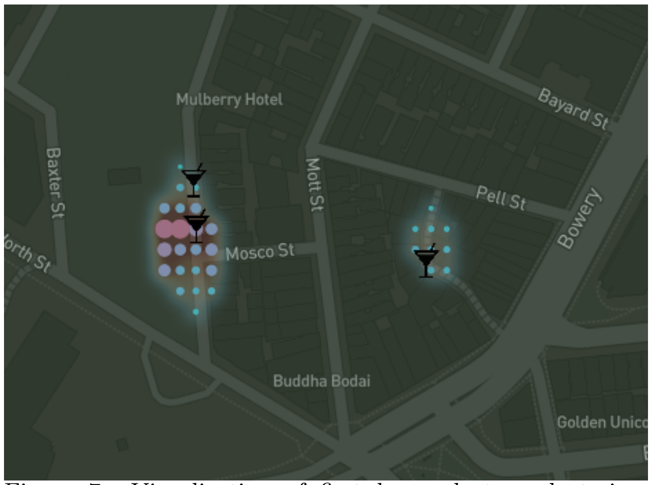


Figure 7: Visualization of first layer clusters clustering around bars in Manhattan.

Score
The second layer clustering results in 350 clusters per year with a high probability to be an interesting location for nightlife. To meet our objective and research question we have to filter these on places being "non-obvious" to the nightlife. This is incorporated by scoring every cluster, showing the likelihood of it being a non-obvious nightlife location, according to the following equation:

$$score(\text{cluster}_i) = data_collection\frac{N^i_{night}}{N^i_{day}} \cdot \left(1.5e^{-0.2N^i_{bars}} + 0.5\right) \tag{1}$$

where:

- i: the ith cluster,
- N^i_{night}: the number of night trips taken in cluster i,
- N^i_{day}: the number of day trips taken in cluster i,
- N^i_{bars}: the number of bars within a Euclidean distance of 0.0005 of cluster i.

.

This equation will result in a score ranging between 0.5 and ∞. A higher score means the cluster is more likely to be a non-obvious nightlife location. A lower score will result in a cluster also being a popular nightlife location however being more obvious.

The score equation consists of two terms:

1. The first term is a ratio showing how much more interesting the spot is at night in comparison to the day.

2. The second term is a penalty term. It will penalize clusters surrounded by more bars, while complimenting clusters having fewer bars in range. The fractions in this term are chosen such that the result of the term will be in the range of 0.5 and 2. 0 bars will double the first term and > 20 bars will halve the first term, showed in Figure 8. The values of these fractions are based on the distribution of the bars over all years showing that the maximum value of bars in range lies around 11. The figure shows for a value of $N^i_{bars} = 11$ the penalty term will return a value slight above 0.5.

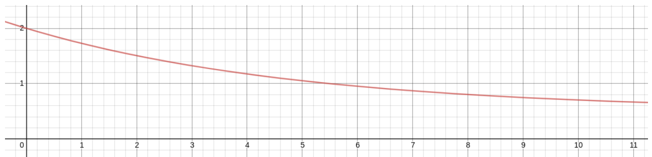


Figure 8: Function of the penalty term: $\frac{3}{2}e^{-\frac{1}{5}N^i_{bars}} + \frac{1}{2}$.

The choice for using Python3 over Spark is Spark not performing well on iterated functions like the greedy clustering algorithm. A possibility would be using only Java, however Python3 has better performance due to the used NumPy allowing for vectorized operations on data. Vectorized operations happen for instance when the distance between cluster is calculated and are significantly faster than sequential Java operations.

5.2.3 *Time series collection*

After scoring all clusters multiple time series per cluster will be collected. This is achieved using Python3 again in combination with NumPy and Pandas. For the time series collection we use Python3 instead of Spark by reason of Spark having problems with the RAM limit. This is due to Spark accumulating old dataframes in memory after joining them with the clusters. Python3 allows more control over the memory usage while also featuring advantageous tools like NumPy and Pandas.

The NumPy array containing the second layer clusters is inserted into a Pandas data frame. Next, the pickup and drop off table for each year are loaded into a Pandas data frame. Hereby only the night trips as defined in table 1 are used. Subsequently, all trips within a Euclidean distance of 0.0005 of each cluster are gathered. Finally, the following time series are collected for each cluster per year using the DateTime information about each trip:

- number of night trips per hour,
- number of night trips per night out of week,
- number of night trips per month,
- number of night trips per year.

Note that we introduce the term night out of week rather than day of week. Since a night out typically spans two consecutive days the night out feature will be represented by the last hours of a day and the first hours of the next day. For instance, Monday night will be the in the time range from Monday night to Tuesday morning.

All Pandas data frames will be converted to a valid GeoJSON file per year which will be used for the visualization. A GeoJSON file for a year would typically look like the one described below, where each element of the features array represent information about one cluster.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "monthStats": {
          "09": {
            "month": [1, ..., 12],
            "numberOfTrips": [...]
          },
          ...
        },
        "nightOfWeekStats": {
          "2009": {
            "nightOfWeek": [1, ..., 7],
            "numberOfTrips": [...]
          },
          ...
        },
        "hourStats": {
          "2009": {
            "hour": [22, 23, 0, 1, 2, 3, 4],
            "numberOfTrips": [...]
          },
          ...
        },
        "yearStats": {
          "year": [2009, ..., 2015],
          "numberOfTrips": [...]
        },
        "numberOfNightTrips": ...,
        "numberOfDayTrips": ...,
        "nightLifeLocationsInNeighbourhood": ,
        "score": ...,
        "geoHash": "..."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [..., ...]
      }
    }
  ]
}
```

Combined, the GeoJSONs are about 1MB per year, which is a size small enough to be visualized in real time on the client side.

5.3 Visualization

The visualization can be found here: `http://178.62.224.151/`

The GeoJSON file is visualized in a web application using Mapbox GL JS, supported in Chrome and Firefox. Using Mapbox GL JS we create a heat map of the clusters with the density of the heat map being linked to the cluster's score. If zoomed in on the heat map the corresponding points for each cluster will appear, the size and color of it resembling its score. Each point is clickable and shows a popup providing information about the cluster, described in the GeoJSON file. Using Chart.JS the time series will be visualized in the popup while also providing other information such as a link to the location on Google Maps.

The menu of the visualization allows to filter between years and displays the top highest scored cluster for the current year.

In order for clusters to be localized conveniently we hash the geographical coordinates to a unique geohash. Using the search bar on the left, clusters can be located using their geohash.

The visualized heat maps show that the found non-obvious hot spots are concentrated in an area which covers North Brooklyn, the bottom half of Manhattan and parts of West Queens. Staten Island and The Bronx possess none of the hot spots we found using the TLC data. These findings shows similarities with the heat maps provided in [1] showing that the nightlife branches discussed in section 4 are also located in these areas.

6. EXPERIMENTS

The second layer clustering results in 2415 clusters that can all be classified as non-obvious nightlife locations, with the likelihood of being non-obvious denoted by the score. In this section we will do a case study of three interesting clusters using the created visualization tool. Besides, we make use of Google Maps Streetview to provided further statements about the locations. Streetview offers the possibility to change photo moments for locations to other periods of time. Using this possibility we are able to see how locations change over time, although the available time moments differ per location.

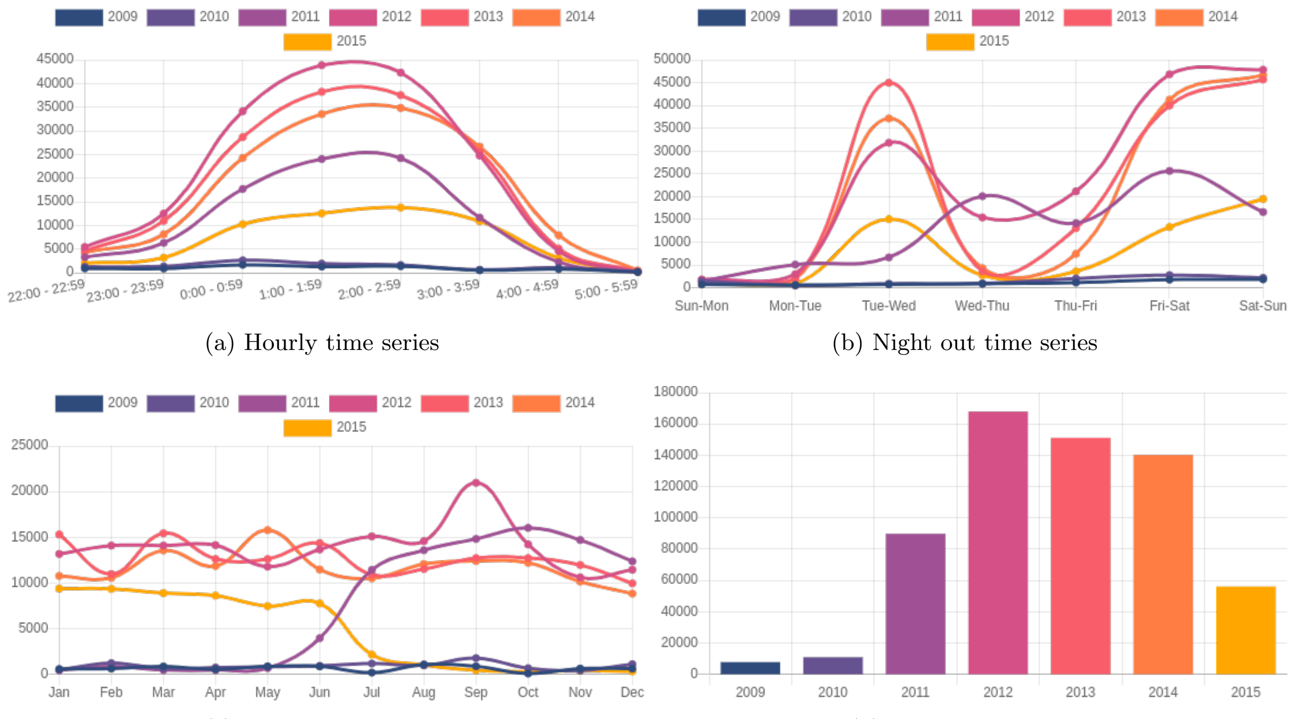
Note that with the provided geohash in the figure of each experiments the corresponding clusters can be found using the search bar in the visualization.

6.1 Gentleman's Club

The first interesting cluster is situated on Clarkson St, of which the statistics are shown in Figure 9. The yearly analysis in Figure 9d shows it being a relatively obscure place in 2009, climbing in popularity in 2011 and 2012 and slowly decreasing after. This is also demonstrated by the hourly analysis in Figure 9a which additionally shows that the peak of taxi trips around this place is reached between 01:00 and 01:59. From the plot in Figure 9b we can gather that the place was mostly visited on Tuesday night and in the weekends. Finally, we observe in Figure 9c that the establishment started becoming popular from June 2011 and beyond while its popularity strongly declines after July 2015.

If we search this location using Google Maps streetview we see a take on the traditional "gentlemen's club" called Westside being situated nearby this cluster until `June2011`. If we progress further in time, to `July2012`, the gentlemen's club looks out of use.

By `August2013` we see a modern looking nightclub Westway appearing on the location of the former gentlemen's club. Next to it has now opened a new "gentlemen's club" called Mystique, since at least `June2014`, accounting for the rise in popularity from 2012 and thereafter. However, the rise in popularity for this hot spot starts from 2011, which



(a) Hourly time series (b) Night out time series

(c) Montly time series (d) Yearly time series

Figure 9: Statistics of taxi trips of a cluster located on Clarkson St, New York (geohash: hfufyyfec).

can be explained by the outside changes taking place later than the change from gentlemen's club to nightclub. This can be supported by the existence of Yelp reviews of WestWay since 2011, `one` of those stating that "the outside sign still says Westside Gentleman's Club".

From May 2016 this location is turned into a construction site, which contributes to its decrease in popularity by taxi trips.

6.2 Club Europe / The Good Room

A second cluster study is done on a cluster belonging to a place on Meserole Ave. On the streetview of Google Maps of `August2014` we see that this cluster is located at the doorstep of a nightclub called Club Europe. The next streetview photo moment is in September 2015 and shows us Club Europe has changed to the nightclub Good Room. The time series of this location are shown in Figure 10.

Figure 10a shows a peak in popularity between 00:00 and 02:00 and Figure 10b shows us the place is mostly visited on weekends which fits the usual visit hours of nightclubs. Even more interesting, Figure 10d illustrates that the place experienced a significant increase in popularity in 2015 which can be related to the switch from Club Europe to the Good Room.

At least one Yelp user posted a `review` supporting this assumption, stating "I can't believe this used to be Club Europa - that was such a dingy venue!". The increase of popularity is already noticeable from the end of 2014, peaking in early 2015 and falling back a little halfway during the year.

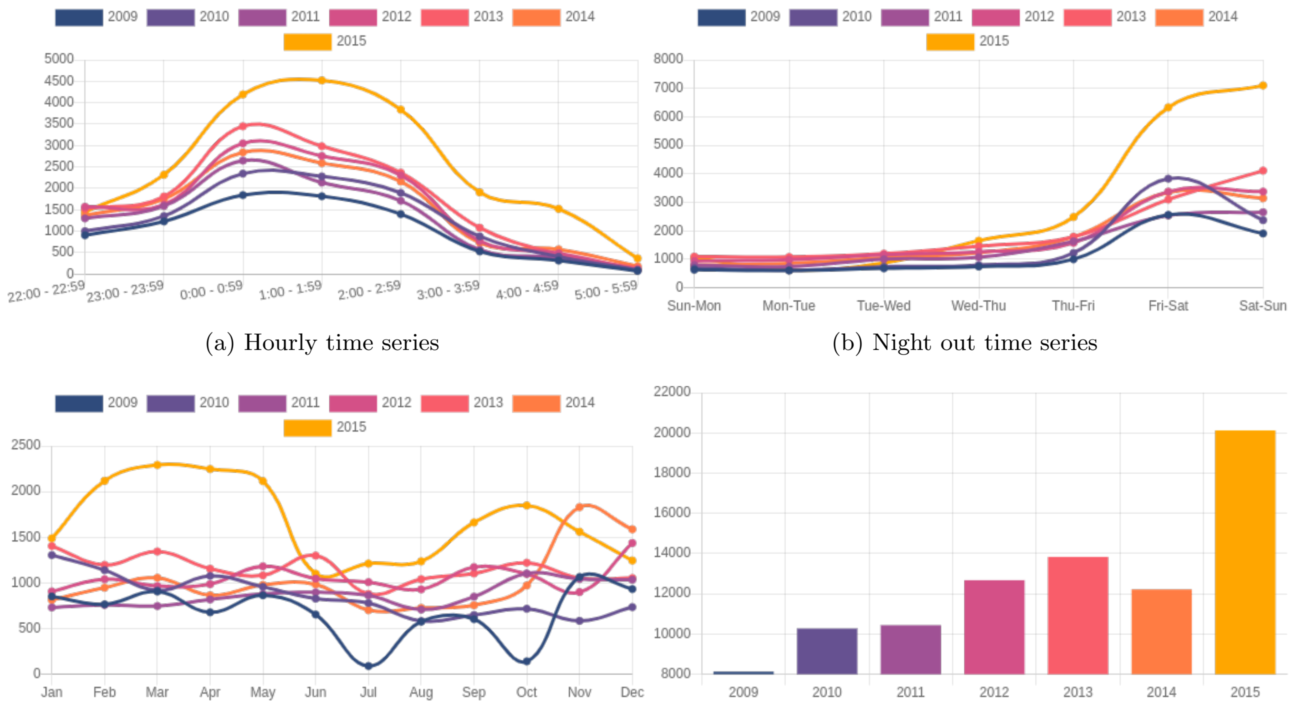
6.3 The NoBar bar

The final cluster we discuss is a cluster seeming to belong to a bar called NoBar, located at Nostrand Avenue. Multiple time series of this location are shown in Figure 11.

Figure 11a shows that the amount of taxi trips is strongly decreasing after 02:00 - 2:59. We can substantiate this with the closing times found online of NoBar which are set to 02:00 for most days. In Figure 11b we can see the bar is slightly more popular on Saturday and Sunday, especially since 2012. Figure 11d demonstrates the cluster reaching it peak of popularity in 2014 which strongly declines after. If we take a look at the Streetview of `July2015` we see NoBar being out of use denoted by the "STORE FOR RENT" sign at the hatch of the bar. A Yelp review posted on `March2015` stating "I love NO Bar! Where did it go?? It's my absolute favorite spot in the neighborhood..." confirms this thought.

7. CONCLUSION

Using the TLC taxi data in combination with the bars dataset we completed our objective. The taxi is over 50 GB and thus had to be converted to much smaller files. This is achieved by first discarding irrelevant information and clustering taxi trips using Apache Spark. Next we use the Kaggle bars dataset to determine the amount of bars close to each cluster, which will later be used in the score of each cluster. The clusters are clustered again, now using Python3 in combination with NumPy. Scoring these new clusters results in a value describing the likelihood of each cluster being non-obvious in the nightlife of NYC. After scoring we collect the time series for every cluster. The time series in combination with other cluster properties will be written into GeoJSON files which are visualized using Javascript's Mapbox GL JS and Chart.js. These files are 1



(a) Hourly time series

(b) Night out time series

(c) Montly time series

(d) Yearly time series

Figure 10: Statistics of taxi trips of a cluster located on Meserole Ave, New York (geohash: hfugqc26r).

MB per year, thus small enough to be visualized in real time on the client side.

The visualization shows the clusters locations on the NYC map while the corresponding time series are plottted for every cluster. Since there are lots of clusters that meet the objective, we have covered three hotpots in section 6. Using a combination of Google Maps Streetview and Yelp reviews we try to explain the cluster's changes in popularity based on taxi trips over time.

Since the answer to our research question is ambiguous we created a map visualization showing over 2000 non-obvious nightlife places in NYC.

7.0.1 Discussion

The weakness of our research lies in three factors: the points described below, the scoring equation and the supplied data.

First, our research has shortcoming since there is little to no literature available which could support us. Therefore we had to make a lot of assumptions which are incorporated in the following points:

1. For each cluster i: $N^i_{night} \geq 100$ in order for it to be evaluated.

2. The maximum range distance bars can be located with respect to a cluster i is set to roughly 50 meter for it to count in N^i_{bars},

3. Cluster i contains all trip locations which lie around it in a $11x8$ meter square,

4. The time ranges to separate N^i_{night} and N^i_{day} time ranges are shown in Table 1.

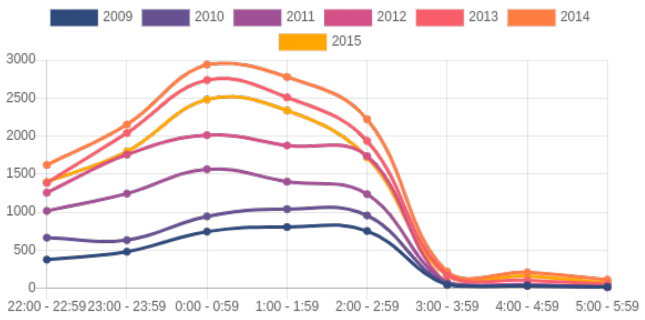
Each of the discussed points is supported with as much literature and research possible, within the short time span of this project. Notwithstanding, it still led to us incorporating assumptions for the integrality of the research. However, all code is written in such a way that the points can be changed easily, possibly leading to a more correct outcome.

The correctness of our result cannot be verified since we do not know the ground truth, thus it is hard to make any comparison. The case studies discussed in section 6 however do give a form of confirmation since all three locations concern a nightlife location. The behaviour of the corresponding time series can also be supported by information found online.

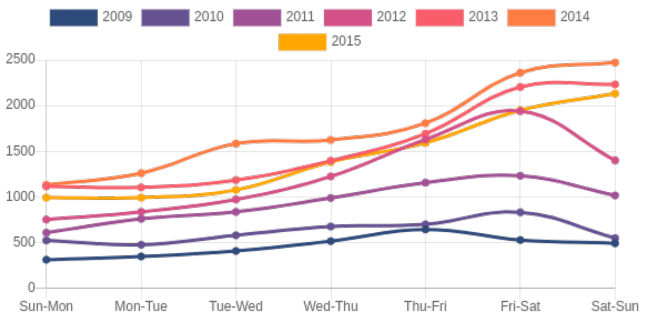
Future work would include more research concerning these points. The research should either confirm our assumptions or provide other values yielding a more correct result.

Second, the scoring equation also shows weaknesses. Looking back, we think we should incorporate the number of trips in the score, rather than only the ratio between the night and day trips. For example, N amount of people at place x at daytime is less interesting when N amount of people is at place x at 02:00. Therefore a weight for both type of trips could have been used in the score mechanism.

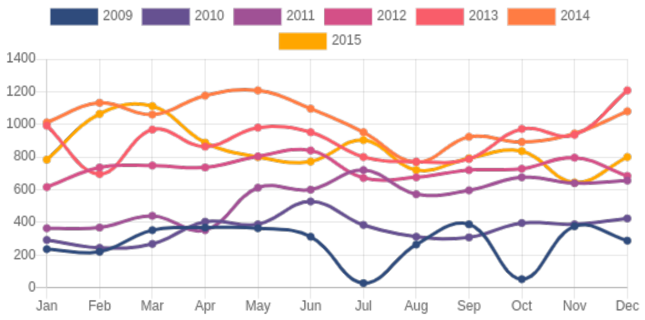
Lastly, the use of the datasets shows some vulnerabilities. The TLC taxi data contains all taxi trips provided by the TLC. Therefore, other providers such as Uber are not taken into account even though they make a big contribution to the taxi traffic of NYC. Thereafter, the taxi zone notation discussed in section 4 has made it less feasible for us to analyse data with this notation. It even led to us ignoring all trip data with this notation which causes a loss of relevance of our research to the present day. The Kaggle bars may also be be incomplete since it only contains bars, clubs and



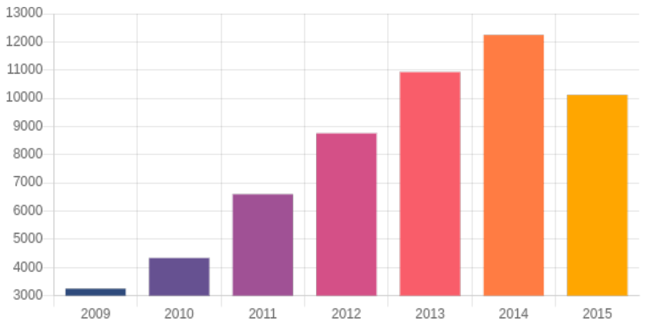
(a) Hourly time series



(b) Night out time series



(c) Montly time series



(d) Yearly time series

Figure 11: Statistics of taxi trips of a cluster located on Nostrand Ave, New York (geohash: hfugm9uej).

restaurants having at least ten registrated noise complaints between 2011 and 2016. Since we are also using taxi trip data from years prior to 2011, the Kaggle bars dataset lacks completeness with respect to our research.

8. REFERENCES

[1] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[2] R. B. Larson. When is dinner? *Journal of Food Distribution Research*, 33(856-2016-57369):38–45, 2002.

[3] Mayor's Office of Media and Entertainment. Nyc's nightlife economy: Impact, assets, and opportunities. Technical report, jan 2019.

Work division

	Work
Jens	Collected data, created first and second layer clusters, converted files to GeoJSONS and collected time series. Wrote parts of the report, focusing on the technical side.
Steven	Wrote and structured majority of the report and corrected all of it. Created all visualization parts. Created presentation slides and structures and wrote most of its text.
Eric	Filtered the data and created the first layer clusters alongside Jens. Did most of the experiments. Wrote parts of the report and corrected spelling and grammar mistakes. Added comments to most of the code.