Engineering NYC PLUTO data for spatio-temporal analysis

This article outlines how to re-engineer the PLUTO data sets from NYC Open Data into usable time-series data.

The article covers five main topics:

- importing and cleaning the raw data;
- re-coding qualitative variables to make them amenable to modeling;
- producing derivative spatio-temporal variables for predictive modeling;
- simple example queries.

This database was engineered for the purpose of spatio-temporal modeling, and there is a separate repository for that analysis. Please visit the analysis repository for more details.

The article is divided into 3 sections:

1. Introduction
2. Data Engineering Overview
3. Simple Queries

Introduction

New York City produces some of the most comprehensive and highest quality public data in the world. The PLUTO data sets, produced by the NYC department of finance are particularly valuable. They were private and very expensive until recently. Now that they have been made public, PLUTO and related data sets could form a valuable well of knowledge for a wide variety of business people and researchers. A particular point of interest is that the PLUTO data sets contain yearly property and land valuations as well as data on new construction and building types.

Unfortunately, while the data sets described above are very valuable, they are segmented into files by year, and *it is not possible directly infer trends over time.* The CSV files from each year have mismatched columns, so it is not as simple as appending the data into one file. Moreover, the files are too big for that approach even if the columns were well behaved. Additionally, the coding schemes used for representing qualitative data about buildings and lots include overlapping categories, making re-coding essential.

Therefore, in order to make the data more usable for researchers and business people, I have assembled the data sets into a database that correctly organizes the records into building identifiers, building properties and time-varying properties. I then briefly demonstrate some of the queries my new database makes possible.

Data Engineering Overview

The PLUTO data set has many variables. A full list can be found in the data dictionaries of each PLUTO release. The most recent release can be found at

> https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-pluto-mappluto.page

and previous releases can be found by selecting the PLUTO data set from the dropdown on the archive site at

> https://www1.nyc.gov/site/planning/data-maps/open-data/bytes-archive.page?sorts[year]=0

However, only certain variables were relevant (and sufficiently complete) for time-series oriented forecasting. They were

- bbl
  - Stands for "borough, block, lot."
  - Is a unique identifier for every property lot in PLUTO.
- YearBuilt
  - Stands for "Year Built"
  - Can be used to compute the age of a building
- YearAlter1 and YearAlter2
  - Stands for "Year Altered 1" and "Year Altered 2," which are years on which a building was gut renovated.
  - Can be used to reconstruct the time since a building has been comprehensively refurbished.
- XCoord and YCoord
  - Stand for "X Coordinate" and "Y Coordinate"
  - Represent the coordinates of the New York and Long Island regional projection used by most GIS software.
  - Captures the location of each lot.
- Block
  - Represents a small scale geo-spatial grouping of neighboring buildings (there are about 9,000 blocks in NYC).
  - Can be used to compute geo-spatial averages of other variables in a small region.
- Zipcode
  - Represents larger neighborhoods that are used for postal routing.
  - Can be used to compute geo-spatial averages of other variables in a large region.
- BuildingClass
  - Stands for "Building Class."
  - Qualitative groupings of buildings by features. Can be used to construct binary building feature variables. Can also be used to filter for residential / rental buildings.
- LotType

  - Stands for "Lot Type."
  - Contains qualitative information about the kinds lots on which buildings reside. Can be used to construct binary building feature variables.
- GarageArea, OfficeArea, CommArea and RetailArea
  - Stand respectively for "Garage Area," "Office Area," "Commercial Area," and "Retail Area."
  - Represents the square footage dedicated to the uses implied by each name. Can be used to construct binary variables indicating whether any space of each type exists in a residential lot.
- BasementCode
  - Stands for "Basement Code."
  - Indicates the kind of basement the primary lot structure has. Can be used to indicate whether a building has a basement.
- NumUnits and NumBldgs
  - Stands respectively for "Number of Units on the lot," and "Number of buildings on the lot."
  - Can be used to compute the number of units per building on a given lot (usually there is only one building, but there are a few thousand lots with multiple buildings—enough to adjust for). Helps clarify the size of a property, irrespective of height.
- Floors
  - Stand for "The number of floors of the tallest building in the lot."
  - Helps distinguish skyscrapers from large, short complexes.
- AsessTot
  - Stands for "Assessed Total Value."
  - For rental properties, before 2019, this value was the taxable value, which was equal to 45% of the market value, but with changes amortized over 5 year delays (for example, a 25% change in value in 2015 would be amortized as a 5% change from 2015-2020, with other amortized values accumulating for each year).
  - At 2019 and later, the variable just represents the raw Annual Market value.
  - Can be used to construct the PCVPU value for forecasting.

2.1 Importing and cleaning the PLUTO data sets In order to manipulate the PLUTO data frames, each file was imported to a postgresql database. Postgresql is an enterprise-level open source relational database management system. It has a command-line interface that acts like a server on a local machine. It was possible to construct the full time-series data set using this local machine mode. The latest version of postgresql can be found at the postgres website. On Unix, the command line interface can be activated using the command

```
sudo -u [user] psql postgres
```

after a user (preferably with admin privelages) is created. It is important to create a user with a name that matches the desktop username so that the special

`\copy` command can be used to export data structures to files. The default `postgres` username will not allow for files to be copied to the desktop file system.

Each .zip file from the PLUTO data set was extracted and the .csv or .txt data files were then imported using the `COPY FROM` command (see `pluto-01--import-files.sql` for the full commands). Since each file had a unique set of columns, each command contained a customized list of variables. Moreover, variable types were sometimes adjusted to deal with formatting issues. Additionally some files were hand edited to remove non-UTF-8 characters using regular expression search. Each file that contained non-standard characters had only one or two non-printing character types that needed to be searched and removed. Additionally, some data sets caused a non-UTF-8 import error even when there were no non-UTF-8 characters present. In these cases, going to the end of the file, pressing return and then deleting the extra line eliminated the problem each time.

Once the data-frames were imported into postgres, it was possible to merge and re-engineer the tables into one large geo-spatial time-series (panel) data-frame.

2.2 Feature Engineering

PCVPU Since the goal of the final model is to predict changes in rents, a response variable was constructed to reflect annual **Percent Changes** across property **Values Per Unit** (assessed total value / total units).

The PCVPU was computed for the intervals ranging from 2005-2006 to 2020-2021. Each interval was named for its respective upper year (2005-2006 -> 2006, etc.). PCVPUs exceeding 300% or below -75% were dropped because these changes only occur if a building is fully gut renovated, or demolished and replaced. Such changes are not relevant to a person contemplating lease renewal terms because tenants cannot be present during complete renovations (full commands can be found in `pluto-03-1-2--truncated_vapue_per_unit.sql`)

Percent change values are problematic for forecasting because all the negative changes are bunched together between negative on and zero while the positive values have a large range. This imbalance makes predicting the magnitude of negative values difficult due to both the non-normality of the distribution and the relative compression of information on one side of the distribution. To overcome these limitaitons, the PCVPU values were symmetrized by taking

$$\frac{1}{1+PCVPU} - 1$$

for the negative values, resulting in a data set that renged from -300% to 300%. These symmetrized data were used for further steps shown below.

Additionally, a discritized version of the PCVPU data were computed for discrete forecasting models.

The PCVPU data were grouped into four categories

below-0%	0%-5%	5%-10%	above-10%
340141	706984	688105	226051

representing negative, small, medium and large percent changes. The reason the entire negative portion of the distribution was grouped together is that land lords tend not to *offer* lower rents to tenants when their property values decrease. Only incoming tenants will see a reduction in offer prices. As a result, any prospective tenant will already see the decrease reflected in a given listing and should not expect to see further rent reductions upon renewal. The positive ranges were chosen to reflect how big a change in rent might *feel* based on personal experience. For example a 5% increase in rent from, say $2,500/mo, is $125 extra per month while a 10% increase would be $250, reflecting the absolute upper bound of what might be considered a "small" and "medium" increase in rents. These numbers, of course, are subjective, but they seemed reasonable given that they split up the data so that the relative frequencies of these events is qualitatively similar to what one might anticipate regarding zero, small, medium and large rent increases (the first and last are less likely while the middle two are equally more likely).

Building Attributes Rental buildings have a variety of qualitative features. The majoirity of these features are implicitly encoded into the "building class" variable. PLUTO classifies residential buildings into mutually exclusive groups according to the following scheme.

Building Class	Count
WALK UP APARTMENTS	
C0 (Three Families)	72,787
C1 (Over Six Families Without Stores)	15,935
C2 (Five to Six Families)	13,620
C3 (Four Families)	16,191
C4 (Old Law Tenements)	2,931
C5 (Converted Dwelling or Rooming House)	2,477
C6 (Cooperative)	3,283
C7 (Over Six Families with Stores)	7,794
C8 (Co-Op Conversion from Loft/Warehouse)	15
C9 (Garden Apartments)	410
S3 (Three Family, One Store or Office)	3,038
S4 (Four Family, One Store or Office)	2,588
S5 (Five or Six Family, One Store or Office)	2,699
ELEVATOR APARTMENTS	
D0 (Co-op Conversion from Loft/Warehouse)	286
D1 (Semi-fireproof, Without Stores)	5,236
D2 (Artists in Residence)	70
D3 (Fireproof, Without Stores)	1,365
D4 (Cooperatives, Other Than Condominiums)	3,536
D5 (Converted)	310
D6 (Fireproof with Stores)	1,204
D7 (Semi-Fireproof with Stores)	2,257
D8 (Luxury Type)	147
D9 (Miscellaneous)	565
TOTAL	**158,727**

Several of the implicit features in this scheme are mutually inclusive. For example, elevator buildings can have or not have store fronts. Other features are exclusive to certain features. For example, only elevator buildings can be classified as fireproofed or semi-fireproofed. If dummy variables were generated for building classes as given in PLUTO, interaction effects of mutually inclusive features would be ignored. Conversely, in some cases, independent effects would be ignored in favor of only interaction effects. For example, buildings with elevators can be fireproof with or without stores. The variable D6 would code for the interaction of being an elevator building, being fireproof and having store fronts, but there would be no code for the independent impact of being an elevator building or having store fronts. Additionally, the building classifications mix the concept of having store fronts with having offices or other commercial units, but does not fully represent all the ways a residential building could have commercial units.

Moreover, there are additional variables that can corroborate or expand the above list of qualities. There are five variables that indicate how much square footage (if

any) is dedicated to (1) retail spaces, (2) other commercial spaces, (3) offices, (4) garages, (5) storage. (1)-(3) can be used, in conjunction with the building classes with stores or offices, to pinpoint which buildings have commercially viable non-residential units. (4) and (5) can be used to code for the existence of garages and storage spaces on a residential lot. Additionally, the lot-type (LotType) variable has classifications that encode whether a building is on a water-front and if it has exposure to the street. Lastly, the basement-code variable (BsmtCode) encodes the existence of building basements. Lastly, the number of units per building in a given block (TotalUnits/NumBldgs) corroborates and expands the information encoded in the building classes regarding the number of units ("families").

The following binary coding scheme was developed to address these issues.

Mutually Inclusive Qualitative Variables

- elevator
  - Means that the lot contains—as its main structure—an elevator building.
  - True if the building class starts with the letter D. False otherwise.
- commercial
  - Means that the residential building contains some units that are used for commercial purposes other than storage or parking.
  - True if the building is of class C7, S3, S4, S5, D6 or D7; or if the building is of class S9 and the number of residential units exceeds 2; or if the ComArea variable is greater than zero; or if the RetailArea variable is greater than zero; or if the OfficeArea variable is greater than zero.
- garage
  - Means the the building has a garage.
  - True if GarageArea exceeds zero. If GarageArea is NULL, it codes as zero.
- basement
  - Means the lot has a basement of any kind.
  - True of BasementCode has the values 1, 2, 3 or 4.
- waterfront
  - Means the lot is located on a waterfront.
  - True if LotType is 2.
- frontage
  - Means the lot abuts at least one street.
  - True if LotType is 1, 3, 4 or 5 (but not 0, 2, etc.)
- cooperative
  - True if the building is managed as a cooperative.
  - True only for buildings classified as C6, C8, D0 or D4.
- converted_loft
  - Indicates if the building was built to be a loft or warehouse, and then

converted for residential use.
  - True only for buildings classified as C8, D0 or D2

Walk-Up Mutually Exclusive Variables

- tenaments
  - Indicates if the property's predominant structure is an Old Law Tenament. These are old buildings that were constructed for poor immigrants. They were originally designed for communal living. Though they have all been renovated to some extent, they can have unusual features like showers in separate rooms from toilets.
  - True only if the building is of class C4.
- garden
  - Indicates if the lot is comprised of a Garden Apartment complex. These are short, wide connected structures, usually containing a large court yard with a garden.
  - True only if the building is of class C9.
- Small walk-up buildings of various sizes are represented by size classes, detailed later.

Elevator-Building Mutually Exclusive Variables

- semi-fireproof
  - Indicates that parts of the structure were constructed to be fireproof, or that the whole structure has been partially fire-proofed. These are usually medium sized apartment buildings.
  - True only if the building is of class D1 or D8.
- fireproof
  - Indicates that the building was constructed to be fireproof. This usually means concrete walls, metal doors and 50s, 60s style nuclear bomb shelters, though glass skyscrapers with sturdy, protected I-beams can also be fire-proof. These are usually mid-sized to large apartment buildings.
  - True only if the building is of class D3 or D6.
- luxury
  - Indicates that the building is a luxury establishment. This means there will be a door-man and many units will be owned by wealthy part-time residents.
  - True only if the building is of class D8.
- artist-in-residence
  - Indicates that a loft or warehouse was converted into special housing for artists. The units are usually very large to accommodate large-scale art construction. They require a special license to occupy and they are rare. There are only 70 artist-in-residence buildings in 2021.
  - True only if the building is of class D2.

- Miscellaneous and converted elevator building classes are implicitly represented by the binary coding scheme outlined above. When a building is classified as an elevator building but nothing else, it is implicitly classified as miscellaneous or converted.

Size Related Variables For small scale structures, differences in size (number of units) play an important role for several reasons. First, there is are structural difference between 3, 4, and 5-6 unit housing, as is evidenced by building classes that encode these distinctions. Second, walk-up buildings with 3-6 and 7-10 units are respectively in different tax classes from other building types. Moreover, the effect of building size on growth in value could be non-linear; though at larger scales, the marginal effect of size per unit is likely to matter less. As a result, it seemed prudent to break buildings up into discrete size-buckets of increasing size.

- "3_unit"
  - Indicates if the principle property on the lot is a 3-unit residential property
  - True if the building is of class C0, S3 or if the the number of total units per building is less than 4.
- "4_unit"
  - Indicates if the principle property on the lot is a 4-unit residential property
  - True if the building is of class C3, S4 or if the number of total units per building is less than five and greater than or equal to four.
- "5-6_units"
  - Indicates if the principle property on the lot is a 5-6-unit residential property
  - True if the number of total units per building is five or six.
- "7_10_units"
  - Indicates if the principle property on the lot is a 7-10-unit residential property
  - True if the number of total units per building is seven to ten.
- "11_25_units"
  - Indicates if the principle property on the lot is a 11-25-unit residential property
  - True if the number of total units per building is eleven to twenty-five.
- "26_50_units"
  - Indicates if the principle property on the lot is a 26-50-unit residential property
  - True if the number of total units per building is twenty-six to fifty.
- "50plus_units"
  - Indicates if the principle property on the lot is a 50+ unit residential property
  - True if the number of total units per building is fifty or more.

One disadvantage of using indicator variables is that the cutoffs lead to arbitrary distinctness at the edges of each bucket (is the difference between a 50 and 51 unit building really so important?). One way to compensate for this shortcoming is to include a scalar representation of building size along-side the indicator variables. In order to meet this need while also capturing shape-related effects, the height of the tallest building on each lot (number of floors) was included. Moreover, since the marginal effect of increasing height is likely to diminish for taller buildings, the quadratic term was encoded (this allows for linear changes in marginal importance).

The full commands for these transformations can be found in the `pluto-02-1--building-features.sql` file.

spatio-temporal PCVPU-derived covariates Several covariates were build from geo-spatial PCVPU signals.

- Autoregressive Data
  - Three years of lagged PCVPU data were computed for each lot and included to capture autoregressive forecasting information. The full commands for this computation can be found in `pluto-03-1-3--truncated_pcvpu_lagged.sql`.
- Spatio-Temporal Data
  - Residential buildings were grouped by block and the average coordinates per block were used to compute the distances between each block (see `pluto-03-2-1--block-distances.sql`).
  - The block distances were used to compute the four nearest neighbors for each block (see `pluto-03-2-2--block-neighbors.sql`).
  - The average PCVPU was computed for each block and organized so that each block contained a record of the PCVPUs for itself and all its nearest neighbors (see `pluto-03-2-3--pcvpu-for-block-neighbors.sql`).
  - Three years of lagged data were computed for each block and its respective neighbors (see `pluto-03-2-4--lagged-pcvpu-for-block-neighbors`)
  - Lagged average PCVPUs were also computed for residential buildings grouped at the zipcode level (see `pluto-03-3--lagged-zipcode-pcvpu.sql`)
  - The above data was then integrated with the autoregressive data by matching each lot's respective block and zipcode to the block and zipcode-level data (see `pluto-03-4--lagged-zipcode-pcvpu.sql`).
  - Additional non-lagged zipcode-level features were also computed (see `pluto-02-2--zip-level-attributes.sql`, though most of them were excluded due to issues with tree depth caused by too many variables)

Simple Queries

Look at changes in average total property values by zip code over the years:

```
SELECT
zipcode
,AVG("2015") as "2015"
,AVG("2016") as "2016"
,AVG("2017") as "2017"
,AVG("2018") as "2018"
,AVG("2019") as "2019"
,AVG("2020") as "2020"
,AVG("2021") as "2021"
FROM assessed_total_value atv INNER JOIN lots b
ON atv.bbl = b.bbl
GROUP BY zipcode
ORDER BY zipcode ASC;
```

Look at changes in aggregate total number of buildings by zip code filtered for manhattan only

```
SELECT
zipcode
,"2015"
,"2016"
,"2017"
,"2018"
,"2019"
,"2020"
,"2021"
FROM (SELECT
zipcode
,SUM("2015") as "2015"
,SUM("2016") as "2016"
,SUM("2017") as "2017"
,SUM("2018") as "2018"
,SUM("2019") as "2019"
,SUM("2020") as "2020"
,SUM("2021") as "2021"
FROM number_of_buildings atv INNER JOIN lots b
ON atv.bbl = b.bbl
GROUP BY zipcode
ORDER BY zipcode ASC) as trends
WHERE zipcode LIKE '100%'
OR zipcode LIKE '101%'
OR zipcode LIKE '102%';
```

It is also possible to use other geo-spatial aggregators like census tract, school district, etc. Moreover, It is possible to filter by building class (for example, using `WHERE building_calss LIKE "C% OR building_calss LIKE "D%"` would return trends for only residential buildings).

I used a similar query as above to produce the following output.

```
 zipcode | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021
---------+------+------+------+------+------+------+------
 10019   | 1199 | 1208 | 1213 | 1207 | 1200 | 1190 | 1188
 10035   | 1494 | 1495 | 1494 | 1521 | 1527 | 1508 | 1504
 10065   | 1220 | 1222 | 1236 | 1243 | 1254 | 1224 | 1220
```

These are time-trends for the zip codes containing Hunter College (10065) and the two apartment buildings I've lived at in New York over the years.