Historical NYC Data Cleaning Process Guide and Dictionary

Dan Miller, Kyi Yeung Goh and Wright Kennedy
Columbia University in the City of New York

August 2019

Acknowledgements

Contents

List of Figures

1 Street Addresses

1.1 Creating a street-enumeration district (ED) dictionary for matching addresses

In order to identify and geocode streets from entries in the census data, street names had to be corrected. In many cases, the entries for street names were either misspelled, misrepresented or left missing. Addressing this issue required us to form a street dictionary that matches original census street entries to correct street names. This dictionary was formed using data from two sources. First, a pull from 1910 geodata and second, ED information from Steve Morse's webpage.

Source 1:*Data from street geography*

The 1910 geodata contained street names along with their respective EDs. This was then concatenated into one long list containing various addresses with EDs. Here, it is useful to note that one address (i.e. Cherry Street) may map onto multiple EDs and hence, will have multiple entries. This same process was also carried out for the Brooklyn street directory.

This pull was cleaned with several changes made in the new *cleaned* column. The following words were standardised to ensure better matches:

- N = "NORTH"
- S = "SOUTH"
- E = "EAST"
- W = "WEST"
- SRT/SR/SST/SEET/TREET/SREET/SRT/REET/ST (only if located at the end of the string) = "STREET" (only if at the end of the string)
- DR/DV/DE/DRV/DRI/DRIV/DRIE = "DRIVE"
- CIR/CRL/CIRC/CR/CL/CIRCL/CICLE = "CIRCLE"
- AVE/AV/AVN/AVEN/AVENU = "AVENUE"
- CT/CRT/CTR/COUR/COT/CORT= "COURT"
- BLVD/BVLD/BV/BLD/BD/BL/BLV = "BOULEVARD"
- ALY/AL/AL/ALLY/ALEY/ALLE/AY = "ALLEY"
- RD/RAD/ROD = "ROAD"
- PL/PLC/PLE/PC/PLAC/PLCE/PCE= "PLACE"
- PKWY/PARKW/PWY/PKW/PRKWY/PKWY/PKW = "PARKWAY"
- PK = "PARK"
- APPR/APR/APPROA/APRCH/APPRCH = "APPROACH'
- TER/TERR/TRC/TRCE/TR= "TERRACE"
- PLZ/PLAZ/PZ/PLZA, = "PLAZA"
- LN/LNE/LAN = "LANE"
- BRG/BRGD/BGE = "BRIDGE"
- HL/HLL/HIL = "HILL"
- 1st to 19th as well as numbers 1-19 (exact matches) were converted into characters

Source 2: Steve Morse ED and Street Data from 1880 and 1910

This was pulled by Dan Miller from Steve Morse's site and there are 4 associated base CSVs which contain an ED-street dictionary for both Brooklyn and Manhattan in 1880 and 1910.

There were concerns that entries in Source 1 was not in itself exhaustive as was the case with a similar dictionary created from the previous pull. This was indeed confirmed with multiple EDs missing and streets missing from EDs that existed in the dictionary. Combinining this was also done in R.

To do so, all 6 associated files were brought into R and their strings handled similarly. They were upper-cased and thereafter, a left join (using the *tidyverse* package) was performed. Conversion from wide to long format was done using the *gather* function. Strings were then cleaned in the same manner to the section about cleaning data from street geography with an additional removal of the backslash that appeared to affect the pull from Steve Morse's site.

1.2 Combining both sources to form a dictionary

Source 1 and 2 were combined to form a dictionary that was utilised to correct street address entries in the census data. Some addresses in the Morse entry did not have suffixes attached (i.e. CHERRY instead of CHERRY STREET). To address this issue, a new column *type* was created to indicate the type of name matched. This includes: street, avenue, drive, circle, court, boulevard, alley, place, parkway, park, approach, terrace, plaza, lane, bridge, hill, heights, hospital, asylum, island, river, jail, river, square, slip, pier and roads. This was done by grouping the cleaned strings from both Source 1 and 2 and filling the empty *type* entries with non-empty types from Source 1.

The dictionary also standardised all streets into a format where the number is followed by the direction as in "100 EAST" rather than having "E 100", "100 E" and other variants of those forms.

There exists another two separate dictionary for usage in the script. They are *"full_BK_dictionary.csv"* and *"full_MN_dictionary.csv"*. These are formatted in a manner to maximise the efficiency of string matching.

1.3 Preparing census for matching: Preprocessing the street address entries in the census data

In addition to the steps above, the census microdata was cleaned with additional steps. This procedure roughly mirrors the script of Logan and Zhang with some key tweaks. To remove issues regarding case-sensitivity, all addresses were raised to upper case. This should not pose a problem as it can be respecified easily. Additionally, the procedure did not consider the threats posed by an "Avenue E" which may be corrected to "Avenue East" given that directional shorthands did not appear to overlap with character names for Manhattan and Brooklyn.

Nonetheless, the same thing which they specified - standardisation of all dictionaries for matching was carried out.

The cleaned strings exist as a separate column. The changes are the following:

- Removing unit numbers that exist in addresses
- Removing remaining ST, ND, RD and TH to allow for better numeric matches

Functions used

1. Rem_dup_word: This is used to remove duplicate words in the census street entry. Many of these included street names such as "Cherry Street Cherry Street" when we wanted to match "Cherry Street".
2. Str_clean: This function repeated what was mentioned in steps 1 to 18 for the census street data.

1.4 Output of cleaning process

In the cleaned addresses dataframe, there are 8 columns in the dataframe:

- ED: This is the initial ED that was given in the pull by Wright.
- Streets: The original street names from the pull
- Cleaned.x: The cleaned version of the pulled names

This pull was cleaned with the **str_clean** function that makes several changes. The following words were standardised to ensure better matches:

- SRT/SR/SST/SEET/TREET/SREET/SRT/REET= "STREET" (only if at the end of the string)
- N = "NORTH"
- S = "SOUTH"
- E = "EAST"
- DR/DV/DE/DRV/DRI/DRIV/DRIE = "DRIVE" (positional matching - "DE" only if it is at the end so streets like "DE KALB" will not be matched)
- CIR/CRL/CIRC/CR/CL/CIRCL/CICLE = "CIRCLE"
- AVE/AV/AVN/AVEN/AVENU = "AVENUE"
- CT/CRT/CTR/COUR/COT/CORT= "COURT"
- BLVD/BVLD/BV/BLD/BD/BL/BLV = "BOULEVARD"
- ALY/AL/AL/ALLY/ALEY/ALLE/AY = "ALLEY"
- RD/RAD/ROD = "ROAD"
- PL/PLC/PLE/PC/PLAC/PLCE/PCE= "PLACE"
- PKWY/PARKW/PWY/PKW/PRKWY/PKWY/PKW = "PARKWAY"
- PK/PRK/PRAK/PAK = "PARK"
- APPR/APR/APPROA/APRCH/APPRCH = "APPROACH'
- TER/TERR/TRC/TRCE/TR= "TERRACE"
- PLZ/PLAZ/PZ/PLZA, = "PLAZA"

- LN/LNE/LAN = “LANE”
- BRG/BRGD/BGE = “BRIDGE”
- HL/HLL/HIL = “HILL”
- HTS/HT/HEIGHT/HEIGHTS/HHT/HEIGT = “HEIGHTS”
- ST = “SAINT” (only if located at the start of the string)
- Place names were also remove such as hotel, hostel, lodge, lodging
- 1st to 19th as well as numbers 1-19 (exact matches) were converted into characters
- Addresses: digits - digits, digits TO digits, digits-digits, removing suffixes such as ST/ND/RD/TH.

Dealing with numbers: - Replaced hundred and two hundred (various spelling) with 1 and 2 - Replaced twenty through to ninety & twentieth to ninetieth (various spelling) with corresponding numbers - Often this results in spaces in numbers and as such, we strip the space between newly created numbers and join them together again - Removed “AND”, “&” and “-” (exact matches)

This was also processed to remove duplicate words in address strings such as “SPRING SPRING STREET”. This cleaned streets in the census then was matched to addresses in the pull that Dan made based on a matching processes that first checked whether the EDs of the census entry which then calls on a list of known streets in those EDs. If this returns a NULL entry then the loop will search through the entire street dictionary for a match which will inevitably result in a poorer match. If this is not null, it will then look for the closest match for streets in those EDs. This is where a proper dictionary containing a possible street name for a given ED is crucial. Poor matching in this case will result from address that are in the ED but for some reason is not included in the dictionary for that particular ED.

This process then produces a dataframe with the following columns after looping through all distinct street name entries for the particular processed batch.

- ED: EDs listed in the census data pull
- House number: House number in the original census data pull (if it exists)
- Streets.X: Original street names strings from the census data pull
- Dwelling serial number| Dwelling serial number 2
- Line number| Line number 2: Line number of the particular entry
- Microfilm page number 3: The microfilm page numbers of the particular entry
- Cleaned.x: The original cleaned street strings from the census addresses.
- ED1: Enumeration district from Dan’s pull, each address maps onto multiple EDs
- ED2: Enumeration district from Dan’s pull, each address maps onto multiple EDs
- ED_streetnames: Original street name from the created ED dictionary.
- corrected_str: The cleaned and matched string based on Jaro-winkler distance
- Dscore: Distance score

- Type: The type of matched string (e.g. road, street, bridge etc.)

The current metric used to flag if this is a good match is currently 0.27. This is somewhat arbitrary but poor matches generally go above this threshold. This was judged based on running our script on a random 10,000 and 100,000 entry sample. The percentage of matches will depend on the addresses but is usually around 20-25% of the total *unique* number of spellings of the addresses.

This is then joined back into the main dataframe and populated according to the specific misspellings noted for those EDs. Prior to this joining, the word "STREET" is re-added to those strings which had the word removed as part of the matching process. This is done by detecting "STREET" in the type column.

1.5 Proposed workflow

1.5.1 Working with the generalised address cleaning script

This entire process of preprocessing and cleaning is streamlined in the cleaning script. There is one for Manhattan and another for Brooklyn.

Here, entries that needed to be cleaned are inputted as a .csv. This script then produces 3 associated files.

1. The list of matched addresses based on distinct street addresses in the census file inputted.
2. The list of flagged matches which requires manual inspection.
3. A complete fill down on the initial .csv inputted with corrected addresses. This is less relevant given that manual correction should occur before the fill-down.

Proposed columns for manual entry

New columns that RAs will fill:

- Manualcorrect: Manual input of correct street names in the same format as the cleaned strings as mentioned above
- ED_correct: Correct ED that the street address in manual correct maps to for that particular entry

To generate after creation:

- Ismanualcorrect: 1 or 0 based on whether string was corrected

1.5.2 Integrating manually corrected addresses into the census data

TO BE UPDATED.

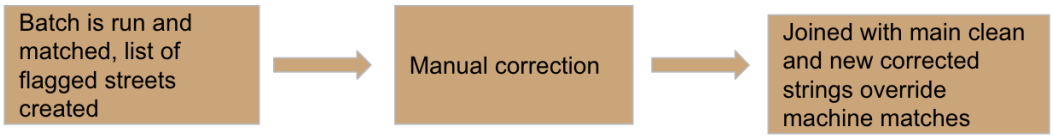

Figure 1: Cleaning workflow visualised

1.6 Current output relevant to street cleaning

1. **RMD FILE** : Generalised address match script
2. **RMD FILE** : Sample address matching script (for testing approach) - replicated over 10k and 100k
3. **RMD FILE** : ED-Street Dictionary script for Manhattan and Brooklyn
4. **.CSV FILE** : List of flagged streets for manual check - SAMPLE
5. **.CSV FILE** : ED-Street dictionary for Manhattan and Brooklyn

2 House Numbers

I approached the house numbers by first filling down the form and extracting observations with distinct house numbers, uncorrected street addresses and household serial numbers. I then mutated a column named original_no to hold the original house numbers in the dataset.

2.1 Dealing with numerical inputs

2.1.1 Dealing with house numbers in street addresses column

However, there existed the possibility of house numbers existing in street addresses which explains some NAs and other problematic entries in the house numbers column. If these existed, I overrided the initial column numbers that were filled in. To find these cases, I used the regex "\d+\ - \|d|\d+\ TO \|d|\d+\-\d*" to look for expressions that contained X - X STREET X TO X STREET or X-X STREET. I did not, however, look for standalone numbers given the risk that street numbers are picked up in place of house numbers.

2.1.2 Dealing with decimals

In many cases, there were house numbers with “1/2” in them. Given that the initial data assigned this a character specification, this was cleaned using regex which replaced that

pattern with .5 and closing any white space that existed.

2.1.3 Dealing with numbers

Here, several steps were taken to obtain columns with specific house numbers.

1. Strings were cleaned by replacing "\b TO \b|\bTO\b|\b & \b|\b&\b|\b AND \b|\bAND\b", indicating ranges with an em-dash.
2. Any blank spaces were also replaced with em-dashes.
3. Slashes were also replaced with em-dashes.
4. All alphabets were removed from the string.
5. All spaces were removed from the string.
6. Looses em-dashes at either the front or end of strings were removed

This was eventually split along em-dashes with each number being separated to denote ranges. Allowances were made for a maximum of three em-dashes "100-1002-1003 Snake Road" would be split into "100", "1002" and "1003" in three different columns.

This is then joined with the main dataframe with cleaned addresses. **As such, it is imperative that the cleaned addresses be as accurate as possible given that they are integrated into the house range matching process.** From this, several columns are selected. They are the extracted house numbers, original street names, dictionary street names, correct street names, the 3 number columns and house words (for strings that appeared alongside the numbers such as REAR, FRONT etc.). A new column called "replaceno" was initiated to check replacements of numbers.

The replacement process occured in three separate segments. The first considered all but the first and last entries. It works on an algorithm that replaces based on the following rules.

First, if cleaned address in this row is the same as the one before and difference between the two numbers is greater than 500 then replace it with the three number columns from the previous rows. This is meant to catch any arbitrary jumps in sequence not due to proper increments and likely stemming from wrong entries. For this replaceno will take the value of "YES" indicating that the number was replaced.

Second, if the cleaned addresses are the same before AND after then take the average if the original entry was NA. For this replaceno will take the value of "YES BETWEEN" indicating that it was replaced based on the averages of numbers before and after.

Third, if after this the streets before is the same as the current one and their difference in numbers exactly match the previous one, suggesting the current one is a 0 then take previous entry numbers. For this replaceno will take the value of "YES" indicating that the number was replaced.

Fourth, if if after this the streets after is the same as the current one and their difference in numbers exactly match the nest one, suggesting the current one is a 0 then take previous entry numbers. For this replaceno will take the value of "YES" indicating that the number was replaced.

In all other cases, the numbers remain the same and a value of "NO" is assigned to replaceno.

The same process is then repeated to help match the first and last entries given that they needed to be taken out in order for the previous matching process to work.

2.1.4 Odd or Even label

Another column, named "oddeven" was added to indicate if the house number was odd or even. This serves as a proxy for the route taken by the enumerator since it signalled the side of the street he or she was on. For this only numbers before the decimal points were matched using a function that identified if the number inputted was even or not. An even number was assigned a value "even", an odd one "odd" and others that are listed as 0 an "NA".

However, another loop is needed to correct for address that have been corrected but have been corrected such that they are no longer on the correct side of the street. Here, the loop looks for, first, if the current row's replaceno column contains "YES BETWEEN" AND whose rows before and after are on the same side of the street AND the current row does not match that of that before. In this case, the first numbers in the string is replaced with that of the previous one and the oddeven column corrected to reflect that change.

Another issues with the previous loop is that it constantly halved the values of NAs since we took averages of entries. To deal with this issue, I first considered the case where an NA is preceded by a non-NA entry whose addressed also matched. These were then replaced with the numbers of those before it IF the addresses matched. Then if the next few entries were also NAs, this will serve as the substitute for the replaced house numbers.

After this process, another loop is used to reassign the odd and even status of the corrected numbers. This process is the same as the previous odd or even assignment mechanism.

2.2 Future work and current weaknesses of approach

1. Addresses - run on large scale to check, see if workflow works for RA. *this is really important since it will be used to match house numbers
2. House numbers and addresses - See if assignment works on large sample. Hard coded logic may be very chunky, there should be a shorter way to hard code the logic. Maybe a different approach altogether - difficult given the number of logics that need to be hard-coded, maybe unsupervised learning?
3. The assignment process for the full sample runs but reduces the sample because it treats all NA as the same and collapses those factors - this issue should not be too challenging to rectify

2.3 Accompanying files for house addresses

1. HNYC Cleaning (House Address script): This is currently run using a distinct sample of the larger sample. To generalise it just remove line 93 in the code.
2. The 100k sample census tract pull.

3 OCR output, potential uses and cleaning progress

This OCR pull contains information about the individual, their marital status, occupation and addresses in a txt format. These are given in a single string which may overflow to a second string due to the length of the entry and the position of it on the original scan from which the OCR derives from.

To clean this, I first began by adding comma after the first full stop to separate the inputs nicely using commas. I then split them accordingly. If both columns that were meant to be filled, column 2 and 3, were empty then assign that column of whether it was a full string or not "1" to indicate that it is not a complete entry. This then runs in a loop that concatenates the string with the string in the previous row and incomplete entry if it is flagged as '1'. Complete cases are then filtered out.

However, there are several issues with this approach.

1. Inconsistent breaks means that the comma separation approach may not work
2. It also means that there will continue to be some columns that remain unmatched
3. Ideally, it will be used to check the range of house numbers to see if the replaced and original numbers make sense. It also helps to make the dataset richer by adding biographical information.

4 References

Lafreniere, D., Gregory, I., & Debats, D. (2017). Routledge Handbook of Spatial History. London: ROUTLEDGE.