Memphis Code Enforcement

Sarah Johnson

2021-07-23

Contents

Chapter 1

About the Author

test

Chapter 2

Introduction

Chapter 3

Literature

Here is a review of existing methods.

Chapter 4

Data: Service Requests Since 2016

Memphis housing code enforcement data from 2016-present is available through Memphis Data Hub. This page uses a dataset downloaded July 19, 2021. At the time of download, there were 53 columns and 1,232,097 rows. R/RStudio was used to view and analyze the dataset.

Each row of the dataset contains all information for a single request, meaning there are not multiple rows for individual cases (except for duplicates).

Despite the large number of records, only a portion are relevant to this paper. This is because the dataset contains **all** requests to 311, not just those related to code enforcement. Additionally, many columns contain duplicate or unhelpful information. For instance, there are 23 columns related to location and under the column `LAST_UPDATED_BY`, every single entry is just the number "460101".

The dataset can be simplified by filtering for code enforcement data and narrowing the number of columns. In the next section I more thoroughly explain why I chose these columns.

NOTE: A list of all column names is available on the Data Hub site linked above, or in R/RStudio enter `colnames(Service_Requests_since_2016)`.

```
CE <- Service_Requests_since_2016 %>%
  filter(DEPARTMENT == "Code Enforcement") %>%
  select(
    INCIDENT_NUMBER, #' unique key; AKA service request (sr) number
    PARCEL_ID, #' parcel ID
    ADDRESS1, #' the street name & number (alt: use FULL_ADDRESS)
    REQUEST_TYPE, #' request category
    CE_CATEGORY, #' category for CE action
```

```
    RESOLUTION_CODE:RESOLUTION_SUMMARY, #' categorizes & explains how the request reso
    REQUEST_STATUS, #' is the request open or closed?
    REPORTED_DATE, #' date the request was reported
    LAST_MODIFIED_DATE, #' date the request was last modified
    OWNER_NAME, #' the assigned CE inspector
    location1 #' geocoordinates
  )
```

The output is 13 columns and 154,844 rows, significantly easier to work with.

4.1 Problems with the dataset

We're ready to begin looking at the data, but before analyzing you should know that **there are significant problems with this dataset**. Though data cleaning is a normal part of research, it feels necessary to explain just how messy this dataset is, and why it's a problem for housing researchers. As someone who enjoys working with and parsing data, this might be the most frustrating dataset I've ever worked with.

Some problems are minor and easily fixed, such as a missing `PARCEL_ID` that can be supplanted with `ADDRESS1`. Other problems are minor and annoying. The field `ADDRESS1` is not meant to contain information like "APT 1", yet "APT" appears 142 times. Sometimes an address will be entered as 123 MAIN and 123 MAIN ST. Sometimes a range of numbers is entered, like 120-123 MAIN ST. Sometimes two or more addresses are entered into the field. With time, these variations can be tidied.

There are many problems with the `DATE` columns. In the original dataset there are eight date columns; I have included two: `REPORTED_DATE` and `LAST_MODIFIED_DATE`. I'll briefly explain why I have omitted the others.

According to the Data Hub website, `REPORTED_DATE` is when a user submits a service requests; the `CREATION_DATE` is an auto-generated date which usually lags a couple hours behind. `NEXT_OPEN_TASK_DATE` is also listed as the same date as `REPORTED_DATE`. This column is not explained on Data Hub, but it is surely related to an inspector's `NUMBER_OF_TASKS`, another column in the original dataset which is also not explained and I did not include here.

`FOLLOWUP_DATE` is confusing and used inconsistently. Sometimes the field has a legitimate date, others times it is NA, and in other cases it is set for 1900-01-01. Sometimes the `RESOLUTION_SUMMARY` will make it obvious that a followup was conducted, but no follow-up date is listed.

Then there's `LAST_MODIFIED_DATE` and `LAST_UPDATE_DATE`, which initial appear to mean the same thing. However, `LAST_UPDATE_DATE` is an auto generated field that seemingly has nothing to do with the case (tens of thousands

of cases were last updated on 2021-06-26 at 05:00:04). `LAST_MODIFIED_DATE` is a user generated field that seems to be the last true human interaction with a case. If a case is closed, the `LAST_MODIFIED_DATE` will *usually* be the same as the `CLOSE_DATE` (it may lag behind a few hours, or more rarely a couple days).

`CLOSE_DATE` and `INCIDENT_RESOLVED_DATE` are duplicate values, though `INCIDENT_RESOLVED_DATE` is not always used. This hints to a larger issue: **inconsistent and vague data entry methods**.

Sometimes there are **multiple values that mean the same thing**, like closed and resolved[1]. The `RESOLUTION_CODE` field in particular has a handful of codes are used for a large umbrella of meanings. For instance, 23% of rows have the code "NJ" for Not Justified. This may mean an inspector has visited the property and did not see a problem, or there was a wrong address[2], or there was a problem and it's been fixed, or there was a problem but it wasn't related to code enforcement; but in most cases there is no further explanation given. There are similar problems for codes tagged CVOM (COMP. V.O. - Miscellaneous, 9% of all cases), CVOID (Closed Void, 4% cases), CO (Closed Other, 3%), and Other (1%). Together, these codes make up 40% of all code enforcement cases.

Another field with this problem is `REQUEST_TYPE`. From 2016 to present, 29% of all requests were simply listed as "Code Miscellaneous".

Some problems are caused by a **lack of updates** on behalf of inspectors. A file may never be closed in the system, even though the inspector is finished looking at the case. There are 779 active cases created in 2018 or earlier (at least 2.5 years old) and it is unclear if some are in a lengthy legal battle or were simply never updated.

There are many many **duplicated entries**. It's hard to determine exactly how many, because the duplicates will have unique values under `INCIDENT_NUMBER` and slightly different date/times. For this reason it is hard to filter out without accidentally omitting multiple legitimate entries under the same property. Also, any entries that were created in error are kept in the system. This leads to inspectors frequently entering "see sr#xxxxxx" in the `RESOLUTION_SUMMARY`, referring to a different `INCIDENT_NUMBER` that contains the correct file.[3] There is also a code specifically for duplicate entries that already have an active file, JA (Justified, Active already file), which has been used 6,588 times, though other codes are known to be used for this same problem.

Other problems are more major. It appears that SeeClickFix (SCF), a program used to allow users to create requests, can cause the **wrong address** to be entered into the system. There is no warning given for this error. When I attempted to sort addresses with the most violations, the top address does not

[1] Resolved has only been used 72 times, which begs the question of whether it needs to exist at all.

[2] INSUF (Insufficient information) is also used for wrong address cases.

[3] There are 6,391 instances of a `RESOLUTION_SUMMARY` mentioning "sr", and nearly all of these rows are likely to be duplicates.

actually exist in the city of Memphis. The inspectors know this and they seem to be able to view the correct address in SCF, and sometimes (but not often) they will manually write this address in the `RESOLUTION_SUMMARY`. It is unclear how many addresses have this problem, but it does affect multiple addresses. As a researcher, I feel nervous conducting research on data with such obvious errors.

Each row is meant to hold all information for a single case. There is also only one column for inspectors to manually enter notes: `RESOLUTION_SUMMARY`. This column has become a **catch-all** for legitimate information, though it is also NA in 23% of rows. In many other cases there is little elaboration, or the `RESOLUTION_CODE_MEANING` is simply repeated. Despite having eight different date columns, it is very common for dates to be entered here with the notes. Because this field is typed, typos are not-uncommon—tenant is spelled "tennant" 51 times, with many other variations such as "tenenat", "tenet", "teneant" and "tenent". Other times a tenant is referred to as a "resident". These variances make it difficult to find all instances of keywords.

Lastly, there are also obvious **privacy issues** included in the field, such as the full name and phone numbers of individuals.

4.2 Data Cleaning

Are we going to keep complaining or try to salvage something from this dataset?

First I determined how many NA values were in each column.

```
colSums(is.na(CE))
```

```
##          INCIDENT_NUMBER                PARCEL_ID                ADDRESS1
##                        0                    15490                     547
##             REQUEST_TYPE             CE_CATEGORY         RESOLUTION_CODE
##                        0                    47089                     547
## RESOLUTION_CODE_MEANING       RESOLUTION_SUMMARY          REQUEST_STATUS
##                      547                    35697                       0
##           REPORTED_DATE       LAST_MODIFIED_DATE               OWNER_NAME
##                        0                        0                    9790
##               location1
##                     1496
```

There are no rows with zero information, but there are some with no *usable* information. I filtered out 20 rows where there is no location, resolution, or written update and placed them in a separate table to double check that I was not omitting useful information.

```
CENA <-
  CE %>% filter(across(
    c(PARCEL_ID,
      ADDRESS1,
      RESOLUTION_CODE,
      RESOLUTION_SUMMARY),
    ~ is.na(.x)
  ))
```

All of these entries are over six months old and most have similar date/times, indicating they were likely duplicates or created in error. After ensuring these rows did not contain viable information, I created a new table omitting them.

```
CENOTNA <- CE %>% filter(if_any(
  c(PARCEL_ID,
    ADDRESS1,
    RESOLUTION_CODE,
    RESOLUTION_SUMMARY),
  ~ !is.na(.x)
))
```

The remaining rows may still have NA values in these columns, but there is at least some other data that may be useful. (**NOTE**: to create a table with no NA values in all selected columns, change `if_any()` to `if_all()` or `across()`)

The remaining NA values fall into two categories: those with no location information, and those missing data from code enforcement.

4.2.1 NA Resolution Code & Resolution Summary

First I created a new table to view rows missing a `RESOLUTION_CODE` and a `RESOLUTION_SUMMARY`.

```
NARCS <- CENOTNA %>% filter(across(c(RESOLUTION_CODE, RESOLUTION_SUMMARY), ~ is.na(.x)))
```

There are 491 rows missing both of these values. I noticed most of the entries were recent, so I filtered the column multiple times by `REPORTED_DATE`.

```
NARCS %>% filter(REPORTED_DATE >= "2021-06-19")
NARCS %>% filter(REPORTED_DATE >= "2021-04-19")
NARCS %>% filter(REPORTED_DATE >= "2021-02-19")
NARCS %>% filter(REPORTED_DATE >= "2020-07-19")
```

Of rows missing both a `RESOLUTION_CODE` and a `RESOLUTION_SUMMARY`, most were created within the past month (368 of 491; 75%), three months (446; 91%), six months (464; 95%), or the past year (476; 97%). It looks like most of these requests are still being worked and thus too recent to have a resolution; let's compare with data on `REQUEST_STATUS`.

```
NARCS %>% count(REQUEST_STATUS) %>% arrange(desc(n))
```

```
## # A tibble: 5 x 2
##   REQUEST_STATUS         n
##   <chr>              <int>
## 1 Open                 424
## 2 In Progress           37
## 3 Back to Department    22
## 4 Closed                 7
## 5 Resolved               1
```

Only 8 rows are listed as closed or resolved (and one entry is clearly a duplicate). It was not immediately clear why these requests were closed with no resolution code, so I manually searched for each address in the `CE` dataset. All addresses had at least one other violation. In two instances, a case was closed less than three minutes after it was created and another request was created at a later date and resolved with more information. In the remaining instances, another violation with more information closed at the same time, indicating the inspector closed multiple entries at once.

```
NARCS1 <- NARCS %>% filter(!str_detect(REQUEST_STATUS, "Closed|Resolved"))
```

4.3 Duplicates & Errors

Searching for duplicates is made difficult by the large number of errors in the dataset. This is made clear when we begin searching for addresses with the most service requests.

```
CENOTNA %>% count(ADDRESS1) %>% arrange(desc(n))
```

```
## # A tibble: 73,772 x 2
##    ADDRESS1               n
##    <chr>              <int>
##  1 <NA>                 527
##  2 746 CHAPEL ST        194
##  3 45 S IDLEWILD ST     117
```

```
##  4 1490 HUGENOT ST       88
##  5 3923 JACKSON AVE      76
##  6 102 PLAINVIEW ST      73
##  7 400 S HIGHLAND ST     72
##  8 1081 COURT AVE APT    71
##  9 3373 STEVE RD         50
## 10 810 WASHINGTON AVE    46
## # ... with 73,762 more rows
```

There are 194 entries listed under the address 746 Chapel St, an address that does not seem to exist (there is a Chapel Rd, but no 746). Reviewing the `RESOLUTION_SUMMARY` reveals the address to be a catch all of errors. Sometimes a different address or a business name is written in the `RESOLUTION_SUMMARY`, indicating the results we are seeing is not the same as what was originally entered, yet code inspectors may be able to view the correct address. It is likely best to remove this address from any analysis involving location.

As for 45 S Idlewild, this is a valid location (an apartment complex) with 117 service requests. A closer look shows three people made 20 requests on February 1, 2019. However, it is not clear if we should trust the information seen in `CREATED_BY_USER`.

At 1490 Hugenot St, there are 117 requests, but this property appears to have the same problem as 746 Chapel. This is revealed by certain entries in the `RESOLUTION_SUMMARY` including "The correct address is xxx Summer Ave." and "INSUFFICIENT INFORMATION WRONG ADDRESS SUBMITTED BY SEECLICKFIX" (SeeClickFix, or SCF, is listed as the request creator for all but five entries at this address).

When we look closer at *who* is making complaints to code enforcement, it at first seems a few people are creating an astounding number of requests. Again, SCF stands for SeeClickFix, an app to create requests.

However, filtering by HELEN.ANDERSON then "tenant" under `RESOLUTION_SUMMARY` shows multiple instances where someone not named "Helen Anderson" is mentioned as the contact person across a wide variety of addresses.

```
test <- CE %>%
  group_by(ADDRESS1) %>%
  summarise(across(c(REQUEST_TYPE, RESOLUTION_SUMMARY), ~ length(unique(.x))))
```

There appear to be many rows that have the same information, with a unique `INCIDENT_NUMBER`. This could indicate a request that was accidentally entered into the system twice.

4.4 Requests by Year

First, how many requests were created each year? We can filter the `REPORTED_DATE` column to find out.

```
CE16 <- CE %>% filter(REPORTED_DATE >= "2016-01-01", REPORTED_DATE < "2017-01-01")
```

Below are the number of requests for each year between 2016 and 2021.

Reported Date	Number of Requests
2016	30,361
2017	26,196
2018	26,846
2019	27,559
2020	26,194
2021 (as of 07/19/2021)	17,688

The number of requests has not changed much over the years.

4.4.1 Active Cases

How many cases are still active from each year? We can find out using the `REQUEST_STATUS` column, filtering out cases marked "Closed" or "Resolved.

```
CE16 %>% filter(!str_detect(REQUEST_STATUS, "Closed|Resolved"))
```

Year	Active Cases
2016	136
2017	111
2018	532
2019	956
2020	1,098
2021	5,630
All Years	**8,463**

There are still 779 active cases from 2018 or earlier.

4.5 Request Type

When a request is entered into 311, it must also be categorized. There are 14 categories for code enforcement data, including 3 categories related to COVID violations which were added in 2020.

The following code was used to find the most common requests for each year:

```
CE16 %>% count(REQUEST_TYPE) %>% arrange(desc(n)) %>% mutate(pct
= n/sum(n)*100)
```

For each year, the vast majority of requests fell into one of five categories, listed below.

Request Type	2016	2017	2018	2019	2020	2021	All	Years
Code Miscellaneous	28%	32%	32%	31%	25%	25%	**45,028**	**29%**
Vehicle Violation	22%	26%	28%	26%	30%	35%	**42,249**	**27%**
Weeds Occupied	20%	18%	17%	19%	14%	11%	**26,161**	**17%**
Junky Yard	18%	13%	11%	12%	12%	15%	**21,283**	**14%**
Substandard, Derelict Structure	9%	8%	10%	10%	7%	9%	**13,778**	**9%**
Sum	**97%**	**97%**	**98%**	**98%**	**88%**	**95%**	**148,499**	**96%**

From 2016 to 2019, more requests were sorted into "Miscellaneous" than any other category (surpassed by vehicle violations in 2020 and 2021).

4.6 Resolution Code

Chapter 5

Discussion

How can so much data say so little? To what end is this great quantity of data if no one can read it? How are we supposed to create a strategic code enforcement if we can't understand the data we already have?

I do not feel comfortable plotting this data because of the large number of errors. It is too easy to misrepresent this data. I do not feel comfortable summarizing this data and drawing conclusions. A map can hide the problem; viewers cannot read all the error messages behind the data. We should not be looking at maps to find where the problems are when our data is the problem.

Instead we should stop, process what we have, find common problems and patterns, and implement improvements. We should be conscious of specific problems faced in the private rental market.

We should do this because our code enforcement's largest hurdle is the **same problem** it faced twenty years ago. Memphis still lacks the ability to efficiently monitor properties and "analyze overall patterns, and hold inspectors accountable for outcomes. At the root of the problem is an outdated computer and data management system. Enforcement decisions remain at the level of individual inspectors: the effectiveness of individual decisions are largely invisible in the absense of systematic analysis and evaluation. Individual inspectors may have sufficient knowledge to understand what is going on with particular properties, and may have an experiential grasp of patterns of property ownership that enables them to know who among property owners is least responsible, for example. If they are hard working and retain a belief that their actions can make a difference, individual inspectors may achieve a reasonable rate of compliance with some violators. But individual efforts, no matter how well intentioned, are by definition *not strategic*." (Betts, 2001, 44)

We have tricked ourselves into thinking that by helping a lot of nonprofits view this data easier, we are being strategic. But we are waiting for the nonprofits

to approach us, and making the data so complicated that they need someone to help them read it.

But **the data itself is the problem**. The lack of reporting is the problem.

Chapter 6

Fixing the Problem

Below I will outline my suggestions.

There's only one column with substance, and even it is lacking.

This dataset only recently passed 449 downloads, even though it's been available for years.

The dataset is lacking obvious information. Rather than 23 columns describing location, it would be appreciated to have info on the actual property. Is there a residential structure? Is it occupied? By owner or tenant? This is basic missing information. Look at how the census organizes housing data.

We can't see the original complaint, only the REQUEST_TYPE and usually we can only see the most recent update, unless the inspector writes multiple updates in the RESOLUTION_SUMMARY.

We can't fix housing neglect if we can't understand the extent of neglect. We won't ever be able to truly understand the extent of neglect until tenants who experience it do not fear eviction. In interviews with multiple professionals who are involved with housing, all mentioned that tenants do not report maintenance problems to code enforcement due to fear of retaliation. They do not believe the system will protect them.

"It's too easy for a landlord to find another excuse to evict," said Sharon Hyde, head of Green and Healthy Homes Initiative's Memphis location.

6.1 Example one

6.2 Example two

Chapter 7

Final Words

We have finished a nice book.

Bibliography

Betts, P. (2001). Best practice number ten: Fixing broken windows – strategies to strengthen housing code enforcement and related approaches to community-based crime prevention in memphis. Technical report.