MethodologyPart1

May 5, 2022

[1]:
```
import sys

# append the directory of law module to sys.path list
sys.path.append('../modules/')
```

[2]:
```
import altair as alt
import arrest
import charge
import law
import pandas as pd
```

1 Cross-city comparisons

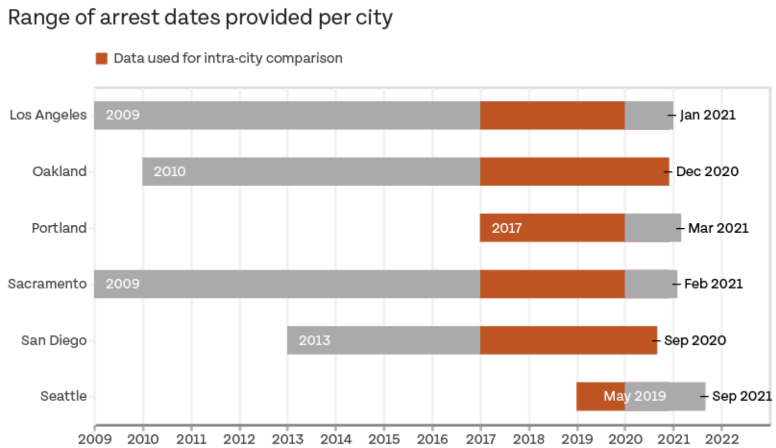
1.1 Decision: Date range

I received data from six cities in time for the story. I'd requested ten years from each, but:

- Portland charged several hundred dollars for even this ~4-year subset.
- Oakland's began with 2010 because we sued for the data and so fulfillment started later than for other cities.
- San Diego could only provide data as far back as 2013, and fulfilled the request earlier than other cities (September 2020).
- Seattle could only provide data as far back as May 2019.

1.1.1 Approach: Compare a subset of "recent" dates

In the radio show, I cited specifics only for Portland, so I used the full range of data the agency provided. I have since compared arrests among cities for **2017 through the end of 2020**, where possible.



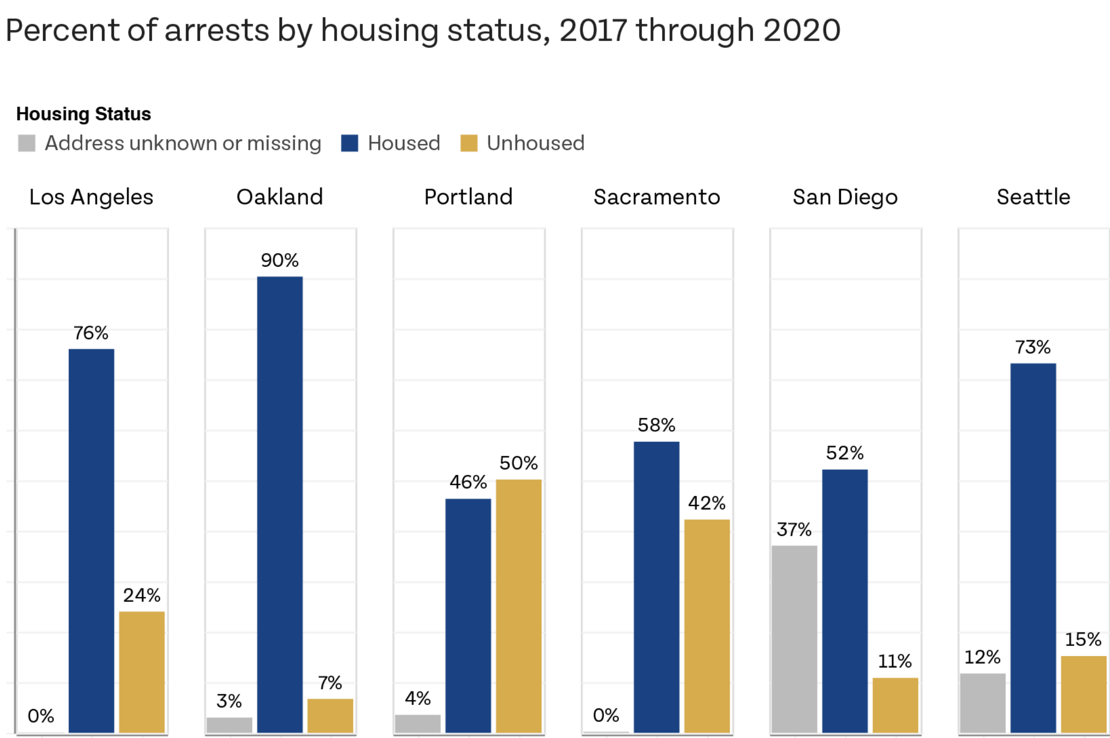
1.1.2 Concerns

San Diego San Diego is short three months (ending September 2020).

- For the purpose of visualization, **my approach is to include San Diego and add a note about San Diego in chart methodologies**.

Seattle

- Seattle is short by much more; with a uniform end date of December 2020, Seattle represents 18 months of data. For the purpose of visualization, **my approach is also to include Seattle and add a note in chart methodologies.**

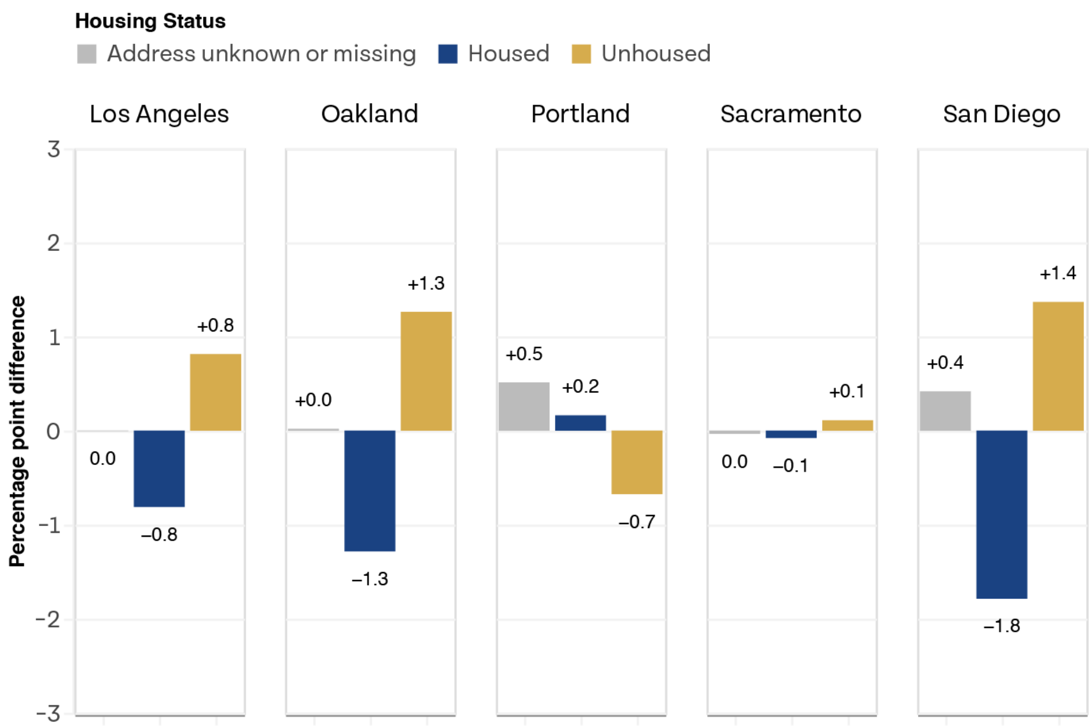


Alternative approaches

- Exclude Seattle altogether, which I would prefer not to do.
- Start analysis at a later date, proceeding through December 2020.
- Reduce analysis to June 2019 through December 2020 for every city, which also presents a problem because of the pandemic.

Percentage point difference of arrests by housing status, 2018 – 2020 compared to 2017 – 2020

Because Seattle data starts in 2019, I'm limiting the analysis to more recent years for each city. Choosing 2017 vs. 2018 affects arrest proportions by housing status by a maximum of 1.8 percentage points (San Diego, "Housed"), who made up an average of 52% of arrests from 2017 through 2020, and 50% of arrests from 2018 through 2020.



1.2 Decision: Juvenile data

Some cities did not provide data on arrests of minors.

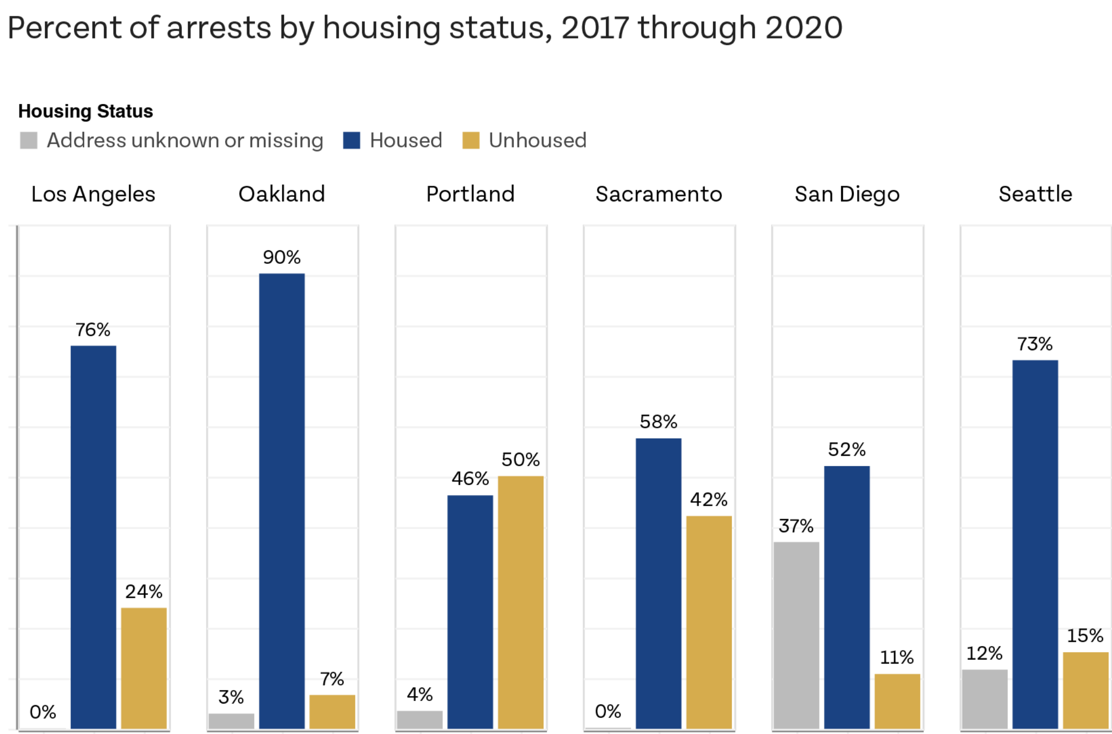
1.2.1 Approach: Exclude all arrests of people who were under age 18 when they were arrested.

1.3 Decision: Categorizing housing status

1.3.1 No address information

Approach: Separate each city into unhoused, housed, and no information. Though I'd requested arrests per se, it appears that Los Angeles also provided citation data, as thousands of entries had no jail booking number attached. Before I excluded these, "no information" arrests made up 28% of LAPD arrests between 2017 and 2020. After I excluded these, "no information" arrests dropped to <1%.

The arrest percentage by housing status chart again:



Concern I followed up with the San Diego Police Department public records administrator about the arrest data I received, and she reiterated that the city did not send citation data.

> The San Diego Police Department has confirmed that the records provided only include arrests as requested.

However, the "no information" proportion of arrests in this data is much higher than in any other city. I asked the same administrator about this April 28th, but as of today (May 5th, 2022), I have not received a response. I also contacted Seattle about its high proportion of such arrests and have also not received a response.

1.3.2 Categorization: Regex

Approach, 'Unhoused' I categorized arrest subjects as unhoused if their recorded address:

```
[3]: regex_df = pd.read_csv('example_data/unhoused_regex.csv', dtype=str)
```

- **was** or **contained**:
  - "homeless" or "transient" or what I deemed to be typos thereof.
    - `0 TRANSIENT`, `299 17TH STREET TRANSIENT`

```
[4]: regex_df[regex_df['_street_address'].str.contains('T[A-Z]+T|H[A-Z]+SS')].head()
```

```
[4]:          city                      _street_address
     0    San Diego                      NONE TRANSIENT
     1      Oakland                           TRAINSENT
     2  Los Angeles                      1942 TRANSUEBT
     3      Seattle  00000 HOMELESS SEATTLE, WA 98104
     4     Portland                            HOMELESS
```

- The name of a social service or emergency shelter

```
[5]: regex_df[regex_df['_street_address'].str.contains('GENERAL')].head()
```

```
[5]:          city                               _street_address
     20     Seattle    1234 GENERAL DELIVERY SEATTLE, WA 98101
     30     Seattle   99999 GENERAL DELIVERY SEATTLE, WA 98105
     46     Seattle  9999 GENERAL DELIVERY BREMERTON, WA 98337
     51     Oakland                           GENERAL DELIVERY
     53  Sacramento                          GENERAL DELLIVERY
```

```
[6]: regex_df[regex_df['_street_address'].str.contains('CITY TEAM')].head()
```

```
[6]:        city     _street_address
     131  Oakland  CITY TEAM SHELTER
```

- The name of or reference to a correctional facility

```
[7]: regex_df[regex_df['_street_address'].str.contains('JAIL|PRISON|RCCC')].head()
```

```
[7]:          city             _street_address
     315  Sacramento           DVI STATE PRISON
     558     Oakland           CONTRA COSTA JAIL
     583  Sacramento          1 CDCRSTATE PRISON
     685     Oakland      SANTA CLARA COUNTY JAIL
     737     Oakland  SAN FRANCISCO COUNTY JAIL
```

- **corresponded to an address of**:
  - a social service or emergency shelter
    - `5130 LEARY SEATTLE` (Ballard Food Bank)

- a government-run social service
    * `2415 W 6TH ST` (LA County Department of Social Services)

Approach, 'Housed' I used regular expressions to find PO Boxes as well, because they're an easy pattern to match and it would save a lot of time and/or money on geocoding services. **I categorized arrests for which addresses were specific PO Box numbers as 'Housed.'**

Concern I can't know what proportion of people with PO Box numbers are actually housed, but I made this decision based on two premises: 1. PO Boxes cost money to reserve (in Portland, the cheapest size is $16 a month and the applicant has to pay for at least three months up front) 2. Applying requires two proofs of identication, one of which "must be traceable to the bearer (prove your physical address)."

1.3.3 Categorization: Geocoding

Data quality I geocoded addresses to more efficiently normalize address fields.

U.S. Census Bureau I geocoded addresses first by attempting to use the free (albeit slow, and less robust) U.S. Census Bureau [geocoding API]. This API returns metadata regarding whether an address matched and, if it matched, whether the match is `exact` or `inexact`. **I used the output of `exact` matches only.**

Geocodio For the second pass, I used Geocodio. Geocodio returns metadata regarding a match's `accuracy type` and `accuracy score`.

Accuracy types, per Geocodio documentation:

> Accuracy types include:
>
> - **rooftop**: on the exact parcel
> - **point**: generally, in front of the parcel on the street
> - **range_interpolation**: generally, in front of the parcel on the street
> - **nearest_rooftop_match**: the nearest rooftop point if the exact point is unavailable
> - **intersection**: An intersection between two streets
> - **street_center**: A central point on the street
> - **place**: zip code or city centroid
> - **county**: county centroid
> - **state**: state centroid

Acccuracy scores:

> Accuracy scores are a reflection of the amount of differences between the input and the output. We generally recommend using results with an accuracy score above 0.8. Results below that threshold can indicate potential issues, such as formatting issues or incomplete addresses.

- **1**: the exact input was returned
- **0.8**: Very close to the input with minor changes made
- **<0.6**: More significant changes made; use these results with caution

I used the following criteria for using outputs:

1. `Accuracy Type` must be `rooftop` or `range_interpolation` **and**
2. `Accuracy Score` must be >=.76

I found upon manual review that addresses were between .76 and .8 when the street names had an edit distance of about two characters, e.g. the input was `123 Brodway` and the output was `123 Broadway`.

Addresses to match on This is an excerpt from California's "HUD 2021 Continuum of Care Homeless Assistance Programs Housing Inventory Count Report." Note that the inventory includes both emergency shelter and permanent housing:

CoC Number: CA-600
CoC Name: Los Angeles City & County CoC

	Family Units[1]	Family Beds[1]	Adult-Only Beds	Child-Only Beds	Total Yr-Round Beds	Seasonal	Overflow / Voucher
Emergency, Safe Haven and Transitional Housing	**3,353**	**10,129**	**10,978**	**0**	**21,107**	**3,409**	**0**
Emergency Shelter	2,821	8,506	8,072	0	16,578	3,409	0
Safe Haven	0	0	400	0	400	n/a	n/a
Transitional Housing	532	1,623	2,506	0	4,129	n/a	n/a
Permanent Housing	**3,500**	**11,434**	**22,122**	**36**	**33,592**	**n/a**	**n/a**
Permanent Supportive Housing*	1,906	5,983	17,694	0	23,677	n/a	n/a
Rapid Re-Housing	1,318	4,545	2,646	0	7,191	n/a	n/a
Other Permanent Housing**	276	906	1,782	36	2,724	n/a	n/a
Grand Total	**6,853**	**21,563**	**33,100**	**36**	**54,699**	**3,409**	**0**

HUD tracks addresses of the service providers in the data that underlies these counts.

[8]:
```python
hic = pd.read_excel(
    '../US/01_inputs/HUD/HIC/2019-Housing-Inventory-County-RawFile.xlsx', dtype=str)
```

But the data is irregular:

[9]:
```python
hic[(hic['HudNum'] == 'CA-600') & (hic['address1'].str.contains('9251'))][
    ['address1', 'city', 'state']
].sort_values(by=['address1'])
```

[9]:
```
                 address1               city state
4950   9251 PIONEER BLVD.  SANTA FE SPRINGS    CA
5473   9251 PIONEER BLVD.  SANTA FE SPRINGS    CA
5475   9251 PIONEER BLVD.  SANTA FE SPRINGS    CA
5476   9251 PIONEER BLVD.  SANTA FE SPRINGS    CA
5477    9251 Pioneer Blvd  Santa Fe Springs    CA
5478    9251 Pioneer Blvd  Santa Fe Springs    CA
```

So I also geocoded all addresses of service providers that operate in the jurisdictions for which I have arrest data. I set another criterion, as well:

[10]:
```
hic[(hic['HudNum'] == 'CA-600') & (hic['address1'].str.contains('9251'))][
    ['Organization Name', 'address1', 'city', 'state', 'Project Type']
].sort_values(by=['address1'])
```

[10]:
```
                                     Organization Name            address1  \
4950  Community Development Commission of the County…  9251 PIONEER BLVD.
5473                                   The Whole Child  9251 PIONEER BLVD.
5475                                   The Whole Child  9251 PIONEER BLVD.
5476                                   The Whole Child  9251 PIONEER BLVD.
5477                                   The Whole Child   9251 Pioneer Blvd
5478                                   The Whole Child   9251 Pioneer Blvd

                   city state Project Type
4950  SANTA FE SPRINGS    CA          RRH
5473  SANTA FE SPRINGS    CA          PSH
5475  SANTA FE SPRINGS    CA          RRH
5476  SANTA FE SPRINGS    CA          RRH
5477  Santa Fe Springs    CA           ES
5478  Santa Fe Springs    CA          RRH
```

One address can correspond to arbitrarily many organizations and, more importantly, greater than one `Project Type`. So after geocoding, I also produced sets of each `Project Type` recorded for an address:

[11]:
```
hic_processed = pd.read_csv(
    '../US/04_outputs/c02_hic_west_coast_geocoded_with_type.csv', dtype=str)
```

[12]:
```
hic_processed[hic_processed['_geocodio_street_address'].str.contains(
    '^9251')][['_geocodio_street_address', '_project_types', '_subcategory',␣
↪'_category']]
```

[12]:
```
     _geocodio_street_address _project_types   _subcategory  _category
3538        9251 PIONEER BLVD   RRH; PSH; ES  mixed support  sheltered
```

Because the above address provides both emergency shelter and permanent supportive housing, **I did not categorize this address as "unhoused."** I did, however, make a note of the subcategory for future reference.

From the set of HIC site addresses, **I categorized each as "unhoused" only if the only recorded Project Type was "ES" (Emergency Shelter)**:

[13]:
```
hic_processed[hic_processed['_category']=='unhoused']['_project_types'].unique()
```

[13]:
```
array(['ES'], dtype=object)
```

[14]:
```python
print(f'Last exported to PDF {pd.Timestamp.now().strftime("%B %d, %Y, ~%H:%M␣
↪PDT")}')
```

```
Last exported to PDF May 05, 2022, ~09:55 PDT
```