PS1

AUTHOR
Huiting Zhang

PUBLISHED
October 5, 2024

1. **PS1:** Due Sat Oct 5 at 5:00PM Central. Worth 50 points. Initiate your **repo**

We use (`*`) to indicate a problem that we think might be time consuming.

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **_HZ_**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **_HZ_** (1 point)
3. Late coins used this pset: **_1_** Late coins left after submission: **_3_**
4. Knit your `ps1.qmd` to make `ps1.pdf`.
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
5. Push `ps1.qmd` and `ps1.pdf` to your github repo. It is fine to use Github Desktop.
6. Submit `ps1.pdf` via Gradescope (4 points)
7. Tag your submission in Gradescope

Read in one percent sample (15 Points)

1.

```
import time
import pandas as pd

start = time.time()

file_path = 'F:\python2\ppha30538_fall2024\problem_sets\ps1\data/parking_tickets_one_percent.c
parking_tickets = pd.read_csv(file_path)

end = time.time()
read = end - start

assert len(parking_tickets) == 287458, f"Expected to have 287458 rows, but got {len(parking_ti

print(f"Time to read file: {read} seconds")
```

```
Time to read file: 1.8747379779815674 seconds

C:\Users\Aurora\AppData\Local\Temp\ipykernel_19484\3830576770.py:7: DtypeWarning:

Columns (7) have mixed types. Specify dtype option on import or set low_memory=False.
```

2.

```
import os
```

```
sample_size_bytes = os.path.getsize(file_path)
sample_size_mb = sample_size_bytes / (1024 * 1024)
full_size_mb = sample_size_mb / 0.01

print(f"Sample size: {sample_size_mb} MB")
print(f"Predicted full dataset size: {full_size_mb} MB")
```

```
Sample size: 80.05409908294678 MB
Predicted full dataset size: 8005.409908294678 MB
```

3.

```
parking_tickets.head()

# The rows on the dataset are ordered or sorted by "issue_date" column.

first_500 = parking_tickets.head(500)
default_order_column = first_500.columns[2]

def is_ordered(column):
    return column.is_monotonic_increasing

column_ordered = is_ordered(first_500[default_order_column])

print(f"Default ordered column: {default_order_column}")
print(f"Whether the column is ordered: {column_ordered}")
```

```
Default ordered column: issue_date
Whether the column is ordered: True
```

Cleaning the data and benchmarking (15 Points)

1.

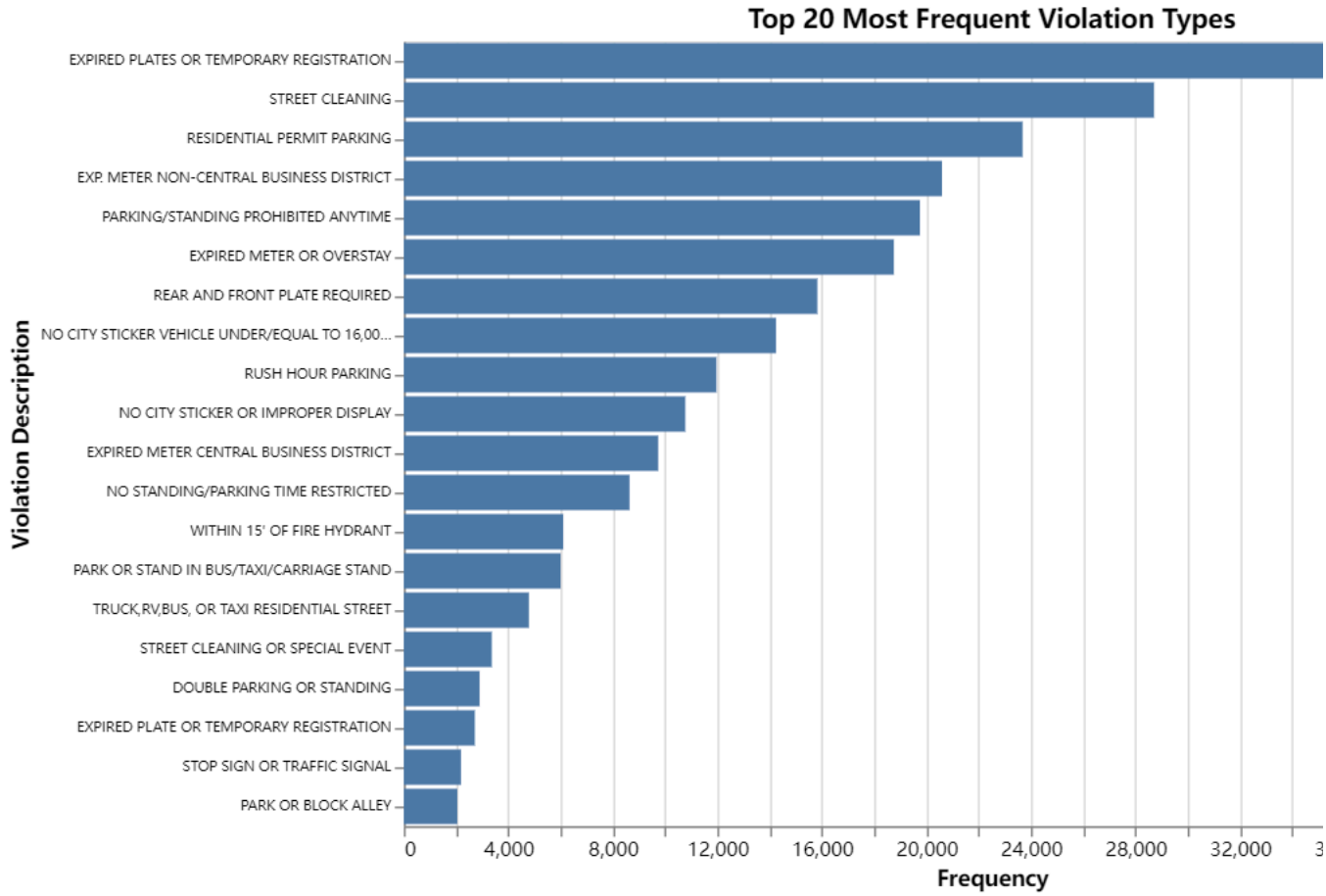
```
parking_tickets['issue_date'] = pd.to_datetime(parking_tickets['issue_date'])

tickets_2017 = parking_tickets[parking_tickets['issue_date'].dt.year == 2017]

sample_tickets_2017 = len(tickets_2017)
print(f"Number of tickets in the sample issued in 2017: {sample_tickets_2017}")

full_tickets_2017 = sample_tickets_2017 * 100
print(f"Estimated number of tickets in the full dataset issued in 2017 : {full_tickets_2017}")
```

```
Number of tickets in the sample issued in 2017: 22364
Estimated number of tickets in the full dataset issued in 2017 : 2236400
```

The number of tickets issued in 2017 in the sample is 22,364, which leads to an estimated 2,236,400 tickets in the full dataset. According to ProPublica, Chicago issues around 3 million tickets annually. This indicates a significant difference, as the estimated 2.24 million tickets are about 25% less than the 3 million reported by ProPublica.

2.

```
import altair as alt

top_violations = parking_tickets['violation_description'].value_counts().head(20).reset_index(
top_violations.columns = ['violation_description', 'count']

alt.Chart(top_violations).mark_bar().encode(
    x=alt.X('count:Q', title='Frequency'),
    y=alt.Y('violation_description:N', sort='-x', title='Violation Description',
            axis=alt.Axis(labelFontSize=7),
          ),
).properties(
    title='Top 20 Most Frequent Violation Types',
    width=600,
    height=400
)
```



Visual Encoding (15 Points)

1.

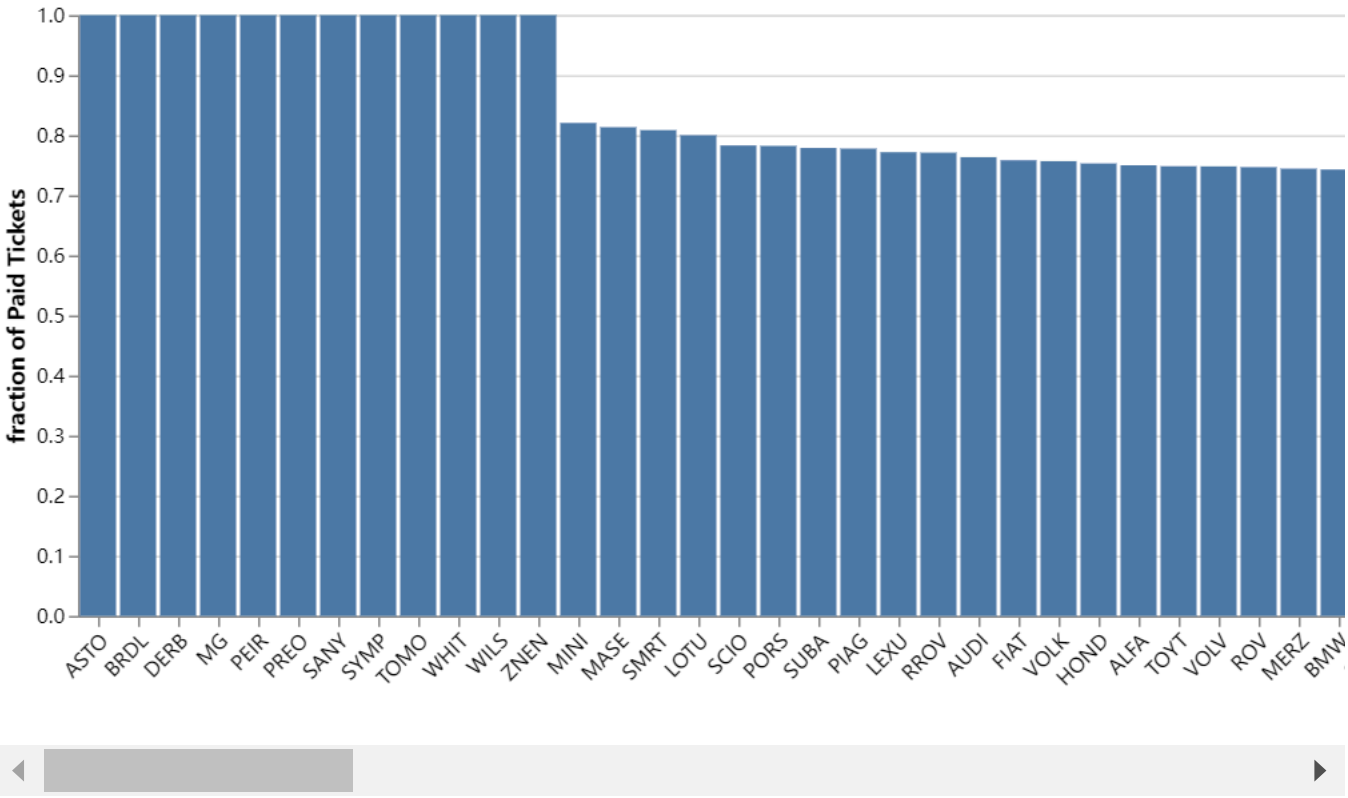
```
data = {
    'Variable Name': [
```

```
        'ticket_number', 'issue_date', 'violation_location', 'license_plate_number', 'license_
        'license_plate_type', 'zipcode', 'violation_code', 'violation_description', 'unit',
        'unit_description', 'vehicle_make', 'fine_level1_amount', 'fine_level2_amount',
        'current_amount_due', 'total_payments', 'ticket_queue', 'ticket_queue_date',
        'notice_level', 'hearing_disposition', 'notice_number', 'officer', 'address'
    ],
    'Altair Data Type': [
        'Nominal', 'Temporal', 'Nominal', 'Nominal', 'Nominal',
        'Nominal', 'Nominal', 'Nominal', 'Nominal', 'Nominal',
        'Nominal', 'Nominal', 'Quantitative', 'Quantitative',
        'Quantitative', 'Quantitative', 'Nominal', 'Temporal',
        'Ordinal', 'Nominal', 'Nominal', 'Nominal', 'Nominal'
    ]
}

df = pd.DataFrame(data)
df
```

	Variable Name	Altair Data Type
0	ticket_number	Nominal
1	issue_date	Temporal
2	violation_location	Nominal
3	license_plate_number	Nominal
4	license_plate_state	Nominal
5	license_plate_type	Nominal
6	zipcode	Nominal
7	violation_code	Nominal
8	violation_description	Nominal
9	unit	Nominal
10	unit_description	Nominal
11	vehicle_make	Nominal
12	fine_level1_amount	Quantitative
13	fine_level2_amount	Quantitative
14	current_amount_due	Quantitative
15	total_payments	Quantitative
16	ticket_queue	Nominal
17	ticket_queue_date	Temporal
18	notice_level	Ordinal
19	hearing_disposition	Nominal
20	notice_number	Nominal
21	officer	Nominal
22	address	Nominal

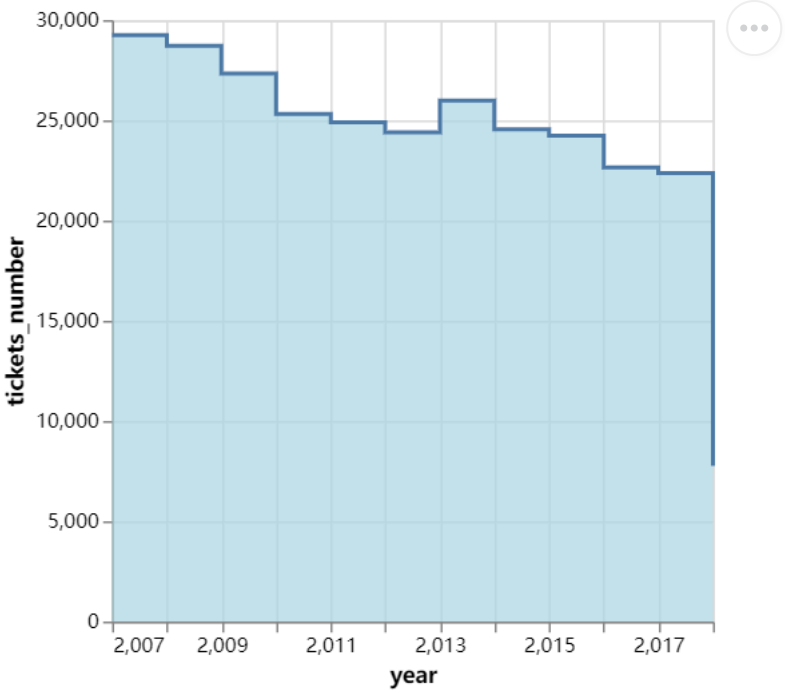
2.

```
paid_fractions = parking_tickets.groupby('vehicle_make').apply(
    lambda x: (x['ticket_queue'] == 'Paid').sum() / len(x)
).reset_index(name='paid_fraction')

alt.Chart(paid_fractions).mark_bar().encode(
    x=alt.X('vehicle_make:N', sort='-y'),
    y=alt.Y('paid_fraction:Q', title='fraction of Paid Tickets'),
).properties(
    title='Fraction of Paid Tickets by Vehicle Make'
).configure_axisX(
    labelAngle=-45
)
```



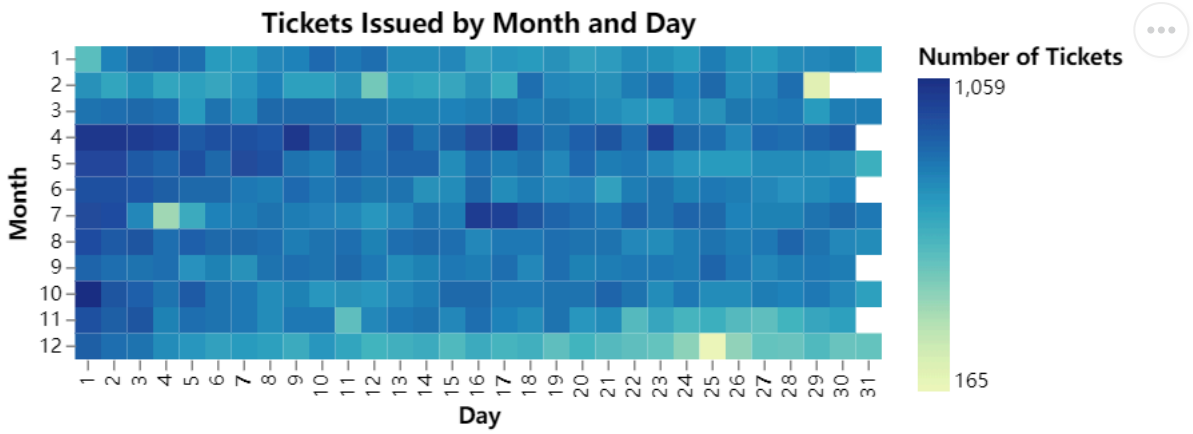
3.

```
parking_tickets['issue_date'] = pd.to_datetime(parking_tickets['issue_date'])

parking_tickets['year'] = parking_tickets['issue_date'].dt.year

tickets_per_year = parking_tickets.groupby('year').size().reset_index(name='tickets_number')

alt.Chart(tickets_per_year).mark_area(
    color="lightblue",
    interpolate='step-after',
    line=True
).encode(
    x='year',
    y='tickets_number'
)
```



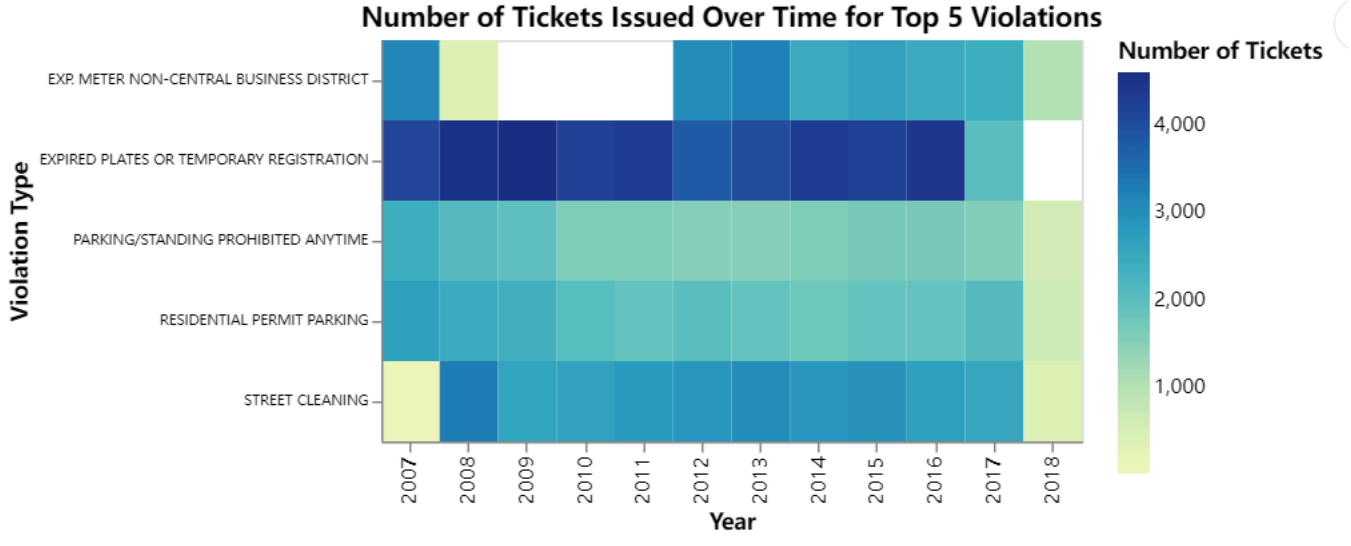
4.

```
parking_tickets['month'] = parking_tickets['issue_date'].dt.month
parking_tickets['day'] = parking_tickets['issue_date'].dt.day

tickets_by_day_month = parking_tickets.groupby(['month', 'day']).size().reset_index(name='tick

alt.Chart(tickets_by_day_month, title="Tickets Issued by Month and Day").mark_rect().encode(
    alt.X('day:O').title("Day"),
    alt.Y('month:O').title("Month"),
    alt.Color('tickets_number:Q').title("Number of Tickets"),
    tooltip=[
        alt.Tooltip('day:O', title="Day"),
        alt.Tooltip('month:O', title="Month"),
        alt.Tooltip('tickets_number:Q', title="Number of Tickets")
    ]
).configure_view(
    step=13,
    strokeWidth=0
).configure_axis(
    domain=False
)
```



5.

```
top_violations = parking_tickets['violation_description'].value_counts().head(5).reset_inde )
top_violations.columns = ['violation_description', 'count']
subset_tickets = parking_tickets[parking_tickets['violation_description'].isin(top_violations[

tickets_over_time = subset_tickets.groupby([subset_tickets['issue_date'].dt.year, 'violation_d

alt.Chart(tickets_over_time).mark_rect().encode(
    alt.X('issue_date:O').title("Year"),
    alt.Y('violation_description:N').title("Violation Type").axis(labelFontSize=7),
    alt.Color('tickets_number:Q').title("Number of Tickets"),
    tooltip=[
        alt.Tooltip('issue_date:O', title="Year"),
        alt.Tooltip('violation_description:N', title="Violation"),
        alt.Tooltip('tickets_number:Q', title="Number of Tickets")
    ]
).properties(
    width=350,
    height=200,
    title="Number of Tickets Issued Over Time for Top 5 Violations"
)
```



6.

Number of Tickets Issued Over Time:

- Pros: It provides a clear view of how the number of tickets issued fluctuates across the years. It is particularly useful for identifying long-term trends or patterns in ticket issuance.
- Cons: It lacks granularity for detailed insights into variations within the year, and it doesn't show any breakdown by violation types or other dimensions, which makes it less informative for understanding specific categories.

Number of Tickets Issued by Month and Day:

- Pros: This plot is great for identifying seasonal trends or patterns in ticket issuance throughout the year. It can easily highlight which months or days have higher activity, so it gives a more detailed temporal analysis than the step chart.
- Cons: It doesn't differentiate between types of violations or provide information about the total number of tickets over time. It focuses on the distribution within a single year but lacks a broader overview.

Lasagna Plot (Tickets by Violation Type Over Time):

- Pros: This plot effectively compares multiple violation types over time, which shows how the prevalence of different tickets has changed. It is helpful for understanding trends specific to certain violation types and provides a good level of granularity by breaking down the data further.
- Cons: It can be a bit harder to interpret compared to the other two due to the complexity of tracking multiple categories. It might also be overwhelming if there are more than five categories to compare.

7.

If the lesson I want the reader to take away is that the enforcement of violations is not evenly distributed over time, the heatmap (tickets by month and day) would be the best plot.

First, the heatmap provides a clear, detailed view of how ticket issuance varies not just by year but also by month and day, making it easy to spot specific periods of higher or lower enforcement activity.

Second, the color gradients in the heatmap naturally draw attention to areas with higher or lower ticket counts, helping to emphasize the uneven distribution of enforcement over time.

Third, by using the heatmap, I can easily demonstrate seasonal or daily patterns that may be less obvious in other types of charts. This would make it clear if certain months or days are more heavily targeted than others.

While the step chart gives a broad yearly view and the lasagna plot shows ticket types over time, neither provides the same level of temporal detail or clarity about the specific days or months when enforcement is higher or lower. Thus, the heatmap is most effective for emphasizing an uneven distribution over time.