30538 Problem Set 3: Parking Tickets Solutions

Peter Ganong, Maggie Shi, and Alex Lan

2026-01-25

Submission and Grading Details

PS3: Due Sat Jan 31 at 5:00PM Central.

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

Background Recap

Read **this** article and **this** shorter article. If you are curious to learn more, **this** page has all of the articles that ProPublica has done on this topic. See the documentation from Propublica's Github on these data **here** and **here**. This problem set is a continuation of PS2 using the same data. Please start by loading the data in the same way as PS2.

Visualization guidelines

To receive full points on visualizations, your visualizations should be coded using `altair`, and should:

- Explicitly declare encodings and data types within `altair`
- Format the graph such that:
  - All axes and units are properly labeled and legible
  - No words or data points are cut off in your compiled PDF
  - Variables should be encoded in a sensible/intuitive way

Data cleaning continued (15 %)

```
import pandas as pd
import altair as alt
import time

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
alt.data_transformers.disable_max_rows()

file_path = 'data/parking_tickets_one_percent.csv'
df = pd.read_csv(file_path, index_col = 0)
```

1. Some of the other articles on the propublica website discuss an increase in the dollar amount of the ticket for not having a city sticker. What was the old violation code and what is the new violation code?

```
# Keep only the rows with violation descriptions that contain 'STICKER'
sticker_violations = df[df['violation_description'].str.contains('STICKER')]
print("All descriptions containing 'STICKER':")
print(sticker_violations['violation_description'].unique())
print("")

# drop rows for over 16,000 pounds
sticker_violations =
 ↪  sticker_violations[~sticker_violations['violation_description'].str.contains('OVER
 ↪  16,000 LBS.')]

# drop rows for improper display of city sticker
sticker_violations =
 ↪  sticker_violations[~sticker_violations['violation_description'].str.contains('IMPROPER
 ↪  DISPLAY OF CITY STICKER')]

print("Remaining descriptions containing 'STICKER':")
print(sticker_violations['violation_description'].unique())
print("")

grouped = sticker_violations.groupby(['violation_description',
 ↪  'violation_code']).size().reset_index(name='n')
print("Comparing violation description and violation code:")
```

```
print(grouped)
print("")

# take the second index because the first is just the start of the data
date_of_change =
 ↪  sticker_violations.groupby('violation_description')['issue_date'].min().iloc[1]

print("Date the fine changed:")
print(date_of_change)
```

```
All descriptions containing 'STICKER':
['NO CITY STICKER OR IMPROPER DISPLAY'
 'NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.'
 'NO CITY STICKER VEHICLE OVER 16,000 LBS.'
 'IMPROPER DISPLAY OF CITY STICKER']

Remaining descriptions containing 'STICKER':
['NO CITY STICKER OR IMPROPER DISPLAY'
 'NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.']

Comparing violation description and violation code:
                              violation_description violation_code      n
0                NO CITY STICKER OR IMPROPER DISPLAY        0964125  10758
1                NO CITY STICKER OR IMPROPER DISPLAY        0976170     15
2  NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 ...       0964125B  14246

Date the fine changed:
2012-02-25 02:00:00
```

From the output above, we can see the three violation codes that are associated with not having a city sticker. While there are three, one has very few observations and so the focus of this question is on the other two. We can see that the code changed in Feb 2012 from "0964125" to "0964125B". We will also give you full credit if you discuss the rare codes "0976170" and "0964125C".

2. How much was the cost of an initial offense under each code? (You can ignore the ticket for a missing city sticker on vehicles over 16,000 pounds.)

```
# use output from table above
sticker_codes = ["0964125", "0976170", "0964125B"]

# get fine amount for first violation by violation code
```

```
grouped = (

 ↪  sticker_violations[sticker_violations["violation_code"].isin(sticker_codes)]
    .groupby('violation_code')['fine_level1_amount']
    .mean()
)
print("Fine amount by code:")
print(grouped)
```

```
Fine amount by code:
violation_code
0964125     120.0
0964125B    200.0
0976170     120.0
Name: fine_level1_amount, dtype: float64
```

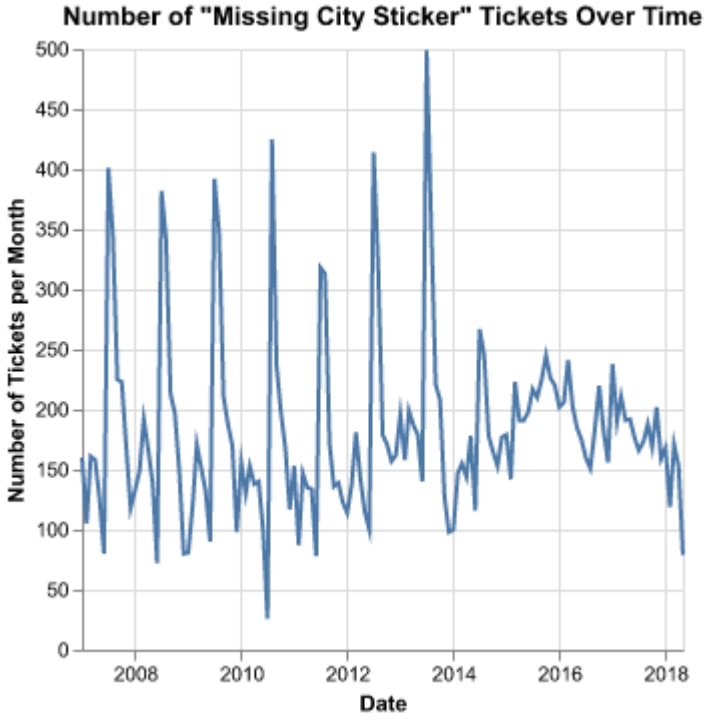
The city switched codes which increased the fine from $120 to $200 for a first offense of not having a city sticker.

Revenue increase from "missing city sticker" tickets (35 %)

1. Using pandas, create a new value for violation codes which combines the two codes that you found in the previous question. Again using pandas, collapse the data to capture the number of "missing city sticker" tickets by month. Then, using Altair, plot the number of tickets over time.

```
# defined above but also here for clarity
sticker_codes = ["0964125", "0976170", "0964125B"]

# create new violation_code variable with all the sticker codes being the
 ↪  same
df.loc[df['violation_code'].isin(sticker_codes), 'new_violation_code'] =
 ↪  "0976170"
df.loc[~df['violation_code'].isin(sticker_codes), 'new_violation_code'] =
 ↪  df['violation_code']

# aggregate data by month
df['issue_date'] = pd.to_datetime(df['issue_date'])
df['yearmonth'] = df['issue_date'].dt.to_period('M').astype(str)
df_monthly = df[df['new_violation_code'] ==
 ↪  "0976170"].groupby('yearmonth').size().reset_index(name='count')
```

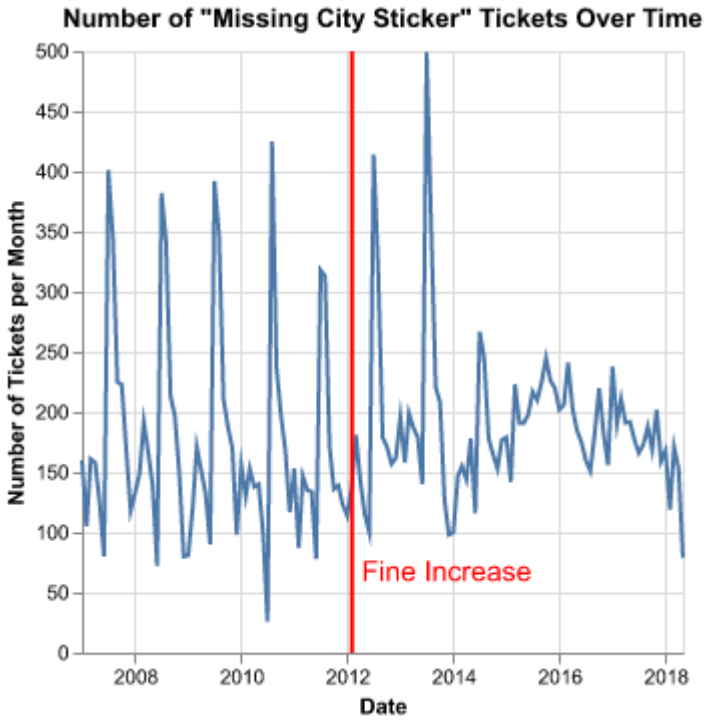
```
# Create the plot
chart = alt.Chart(df_monthly).mark_line().encode(
    x=alt.X('yearmonth:T', title='Date', axis=alt.Axis(format='%Y')),
    y=alt.Y('count:Q', title='Number of Tickets per Month')
).properties(
    title='Number of "Missing City Sticker" Tickets Over Time'
)
chart
```



2. Suppose that your reader wants to be able to use the plot to deduce when the price increase occurred. Add frequent or custom date labels on the x-axis of your plot such that the date of the price increase is readily apparent. We haven't covered Altair's date labeling features in class so you'll first need to find the relevant help page in the documentation. Which help page did you use?

```
# convert yearmonth to a string so we can compare it to date_of_change, and
 ↪  use this to identify where to put the annotation
df_monthly['yearmonth'] = pd.to_datetime(df_monthly['yearmonth'],
 ↪  format='%Y-%m')
df_monthly['yearmonth'] =
 ↪  df_monthly['yearmonth'].dt.to_period('M').astype(str)
```

```
# ensure date_of_change is a datetime object before using strftime
date_of_change_dt = pd.to_datetime(date_of_change)
month_of_change_str = date_of_change_dt.strftime('%Y-%m')

# create the annotation
annotation = alt.Chart(df_monthly[df_monthly['yearmonth'] ==
 ↪  month_of_change_str]).mark_text(
    align='left',
    baseline='bottom',
    dy= 50,
    dx=5,
    fontSize=14,
).encode(
    x=alt.X('yearmonth:T'),
    y=alt.Y('count:Q'),
    text=alt.value('Fine Increase'),
    color = alt.value('red')
)

# create the vertical line
vertical_line = alt.Chart(pd.DataFrame({'yearmonth':
 ↪  [month_of_change_str]})).mark_rule(
    color='red',
    strokeWidth=2
).encode(
    x='yearmonth:T'
)

# combine the chart, annotation, and vertical line into the final chart
final_chart = chart + annotation + vertical_line

final_chart
```



The **Text** and **Rule** pages of the documentation are useful.

3. The City Clerk said the price increase would raise revenue by $16 million per year. For now, ignore the fact that many tickets are not paid and assume that the number of tickets issued is the same before and after the policy change. Using only the data available in the calendar year prior to the increase, how much of a revenue increase should they have projected? Remember that you are working with a one percent sample of the data. Assume that the number of tickets of this type issued afterward would be constant and you can assume that there are no late fees or collection fees, so a ticket is either paid at its face value or is never paid.

```
# get the number of sticker tickets given in 2011
sticker_ticket_count_2011= (df
    .loc[df['issue_date'].dt.year == 2011]
    .loc[df['new_violation_code'] == '0976170']
    ['ticket_number'].count()
)
# remember to multiply by 100 to get full sample
print(f"Number of sticker tickets in 2011: {sticker_ticket_count_2011 *
 ↪  100:,.0f}")

print(f"Naive Estimate for Increased Revenue: ${(sticker_ticket_count_2011 *
 ↪  100 * (200 - 120)):,.2f}")
```

```
Number of sticker tickets in 2011: 193,500
Naive Estimate for Increased Revenue: $15,480,000.00
```

There were a bit under 200k sticker tickets issued in 2011. Under the assumptions listed above, increasing the ticket cost from 120 to 200 would have increased revenue by about $15.5M which is close to $16M.

3. What happened to repayment rates (percentage of tickets issued that had payments made) on this type of ticket in the calendar year after the price increase went into effect? Suppose for a moment that the number of tickets issued was unchanged after the price increase. Using the new repayment rates in the year after the price increase occurred, what would the change in revenue have been?

```
df_2011 = (
    df
    .loc[df['issue_date'].dt.year == 2011]
    .loc[df['new_violation_code'] == '0976170']
)
df_2013 = (
    df
    .loc[df['issue_date'].dt.year == 2013]
    .loc[df['new_violation_code'] == '0976170']
)

repayment_rate_2011 = df_2011[df_2011['ticket_queue'] == 'Paid'].shape[0] /
 ↪  len(df_2011)
repayment_rate_2013 = df_2013[df_2013['ticket_queue'] == 'Paid'].shape[0] /
 ↪  len(df_2013)

print(f"Repayment rate in 2011: {round(repayment_rate_2011, 2)}")
print(f"Repayment rate in 2013: {round(repayment_rate_2013, 2)}")

# Assuming the number of tickets issued was unchanged
naive_revenue_2011 = sticker_ticket_count_2011 * 100 * 120 *
 ↪  repayment_rate_2011
naive_revenue_2013 = sticker_ticket_count_2011 * 100 * 200 *
 ↪  repayment_rate_2013

change_in_revenue = naive_revenue_2013 - naive_revenue_2011

print(f"Change in revenue: ${change_in_revenue:,.0f}")
```

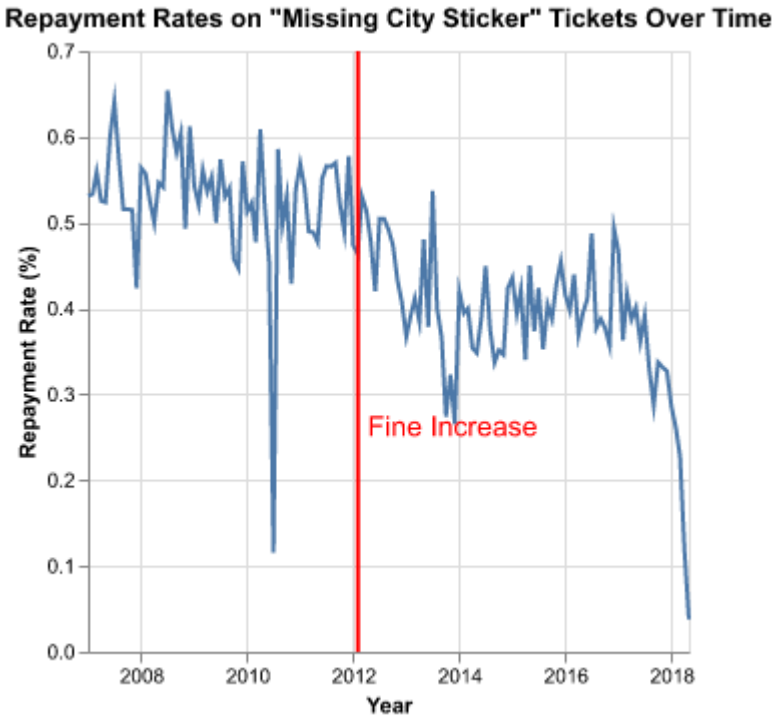
```
Repayment rate in 2011: 0.54
Repayment rate in 2013: 0.41
Change in revenue: $3,181,155
```

After the new rule went into effect, the repayment rate for these tickets dropped from 54% to 41%. If we recalcuate the increase in revenue taking into account the fact that not all tickets are paid, we find that revenue increases by about $3.2M.

5. Make a plot with the repayment rates on "missing city sticker" tickets and a vertical line at when the new policy was introduced. Interpret.

```
# calculate repayment rates by year
repayment_rates = (df
    [df['new_violation_code'] == '0976170']
    .assign(ticket_payed = df['ticket_queue'] == 'Paid')
    .groupby(df['yearmonth'])
    .agg(repayment_rate=('ticket_payed', 'mean'))
    .reset_index()
)

# create the chart
chart = alt.Chart(repayment_rates).mark_line().encode(
    x=alt.X(
        'yearmonth:T',
        title='Year',
        axis=alt.Axis(format='%Y')
    ),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate (%)')
).properties(
    title='Repayment Rates on "Missing City Sticker" Tickets Over Time'
)

# convert yearmonth to a string that can be compared to to
 ↪  month_of_change_str for the annotation
repayment_rates['yearmonth'] = pd.to_datetime(repayment_rates['yearmonth'],
 ↪  format='%Y-%m')
repayment_rates['yearmonth'] =
 ↪  repayment_rates['yearmonth'].dt.to_period('M').astype(str)
# ensure date_of_change is a datetime object before using strftime
date_of_change_dt = pd.to_datetime(date_of_change)
month_of_change_str = date_of_change_dt.strftime('%Y-%m')

# create the annotation
```

```
annotation = alt.Chart(repayment_rates[repayment_rates['yearmonth'] ==
 ↪  month_of_change_str]).mark_text(
    align='left',
    baseline='top',
    dy= 80,
    dx=5,
    fontSize=14,
).encode(
    x=alt.X('yearmonth:T'),
    y=alt.Y('repayment_rate:Q'),
    text=alt.value('Fine Increase'),
    color = alt.value('red')
)

# combine chart, vertical line, and new annotation
chart + vertical_line + annotation
```



Repayment rates were falling a bit over time but appear to have fallen more sharply during the two years after the price increase.

6. Suppose that the City Clerk were committed to getting more revenue from tickets. What

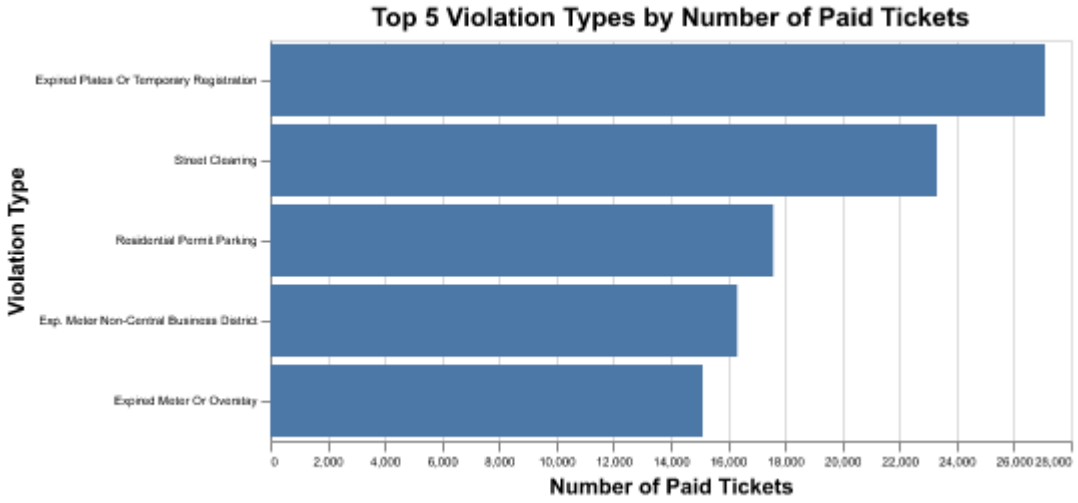
three violation types would you as an analyst have recommended they increase the price of? Consider both the number of tickets issued for each violation type and the repayment rate for each violation type. You may assume there is no behavioral response to price changes (i.e. people continue to commit violations at the same rate and repay at the same rate). Make a plot to support your argument and explain in writing why it supports your argument.

We assume that the number of tickets issued and repayment rates remain constant after a price change. The answer depends on whether the City Clerk increases prices by a fixed dollar amount or by a percentage.(either is fine for completion).

Case 1: Fixed dollar increase. If we increase prices by a fixed dollar amount Δ, then the additional revenue from each violation type equals (number of paid tickets) $\times$ Δ. Therefore, we should target violations with the most paid tickets.

```
# calculate number of paid tickets for each violation type
tickets_by_code = (df
    .groupby(['violation_description'])
    .agg(
        n =('ticket_queue', 'count'),
        n_paid=('ticket_queue', lambda x: (x == 'Paid').sum()),
        avg_fine1=('fine_level1_amount', 'mean')
    )
    .sort_values('n_paid', ascending=False)
    .reset_index()
    .head(5)
)

# clean up violation description to be in title case, not upper case
tickets_by_code['violation_description'] =
 ↪  tickets_by_code['violation_description'].str.title()

# create a horizontal bar chart, sorted by number of paid tickets
chart = alt.Chart(tickets_by_code).mark_bar().encode(
    y=alt.Y('violation_description:N', title='Violation Type', sort='-x'),
    x=alt.X('n_paid:Q', title='Number of Paid Tickets')
).properties(
    title='Top 5 Violation Types by Number of Paid Tickets',
    width=400,
    height=200
).configure_axis( # Configure the axis so that the labels are not cut off
    labelFontSize=6,
    labelLimit=400
```

```
)

chart
```

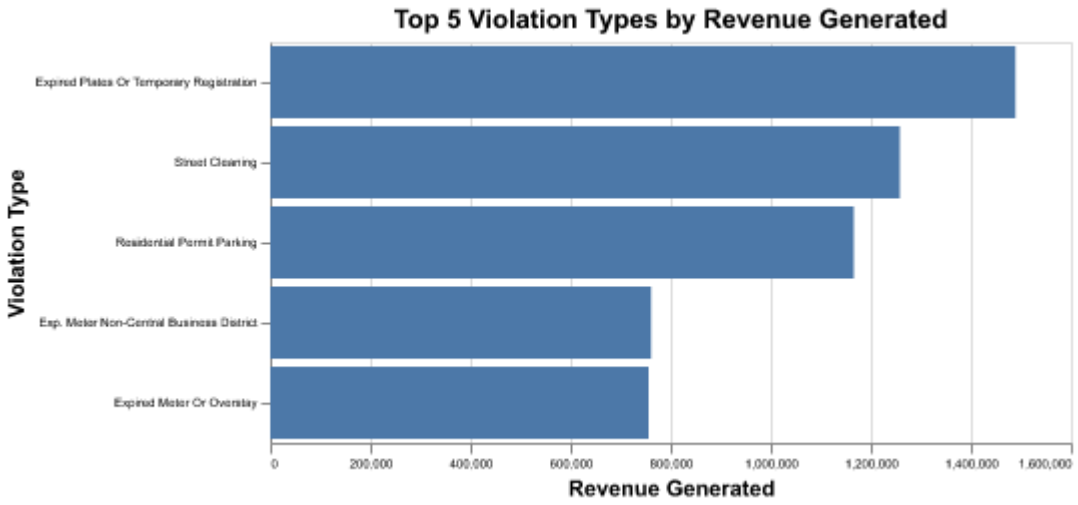


The top three are expired plates or temporary registration, street cleaning, and residential permit parking.

Case 2: Percentage increase. If we increase prices by a percentage p, then the additional revenue from each violation type equals (current revenue) $\times$ p. Therefore, we should target violations with the highest current revenue.

```
# calculate revenue for each violation type
tickets_by_code['revenue'] = tickets_by_code['n_paid'] *
 ↪  tickets_by_code['avg_fine1']

chart = alt.Chart(tickets_by_code).mark_bar().encode(
    y=alt.Y('violation_description:N', title='Violation Type', sort='-x'),
    x=alt.X('revenue:Q', title='Revenue Generated')
).properties(
    title='Top 5 Violation Types by Revenue Generated',
    width=400,
    height=200
).configure_axis(
    labelFontSize=6,
    labelLimit=400
)
```

```
chart
```



The top three revenue-generating types are the same as above: expired plates or temporary registration, street cleaning, and residential permit parking.

Headlines and sub-messages (25 %)

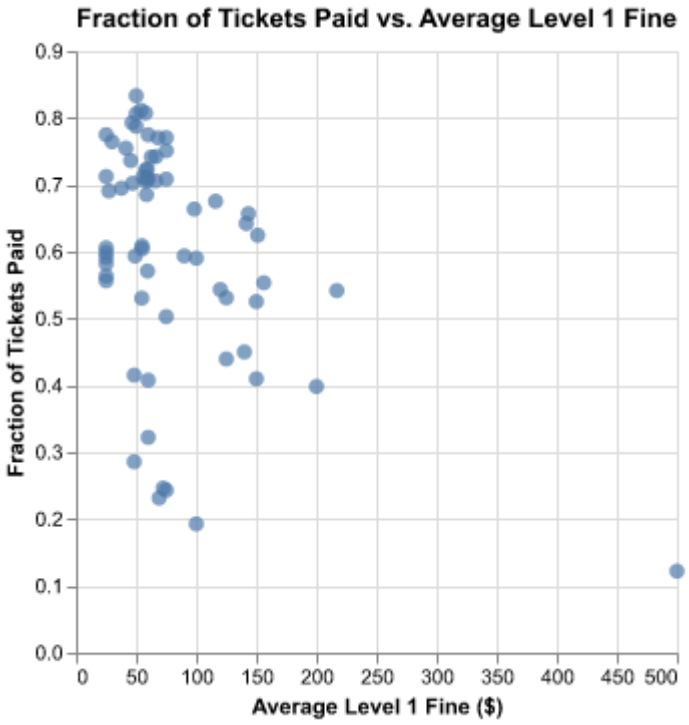
1. The City Clerk has now begun to wonder… maybe raising ticket prices will lead to a decline in repayment rates after all. Make a data frame where each row is a violation description, the fraction of time that the ticket is paid, and the average level 1 fine. Sort this dataframe based on how many total tickets of each type have been issued. Print the rows for the 5 most common violation descriptions.

```
repayment_rate_by_code = (df
    .assign(ticket_paid = df['ticket_queue'] == 'Paid')
    .groupby(['violation_description'])
    .agg(fraction_paid=('ticket_paid', 'mean'),
        average_level1_fine=('fine_level1_amount', 'mean'),
        n_tickets=('ticket_number', 'count'))
    .sort_values('n_tickets', ascending=False)
    .reset_index()
)
print(repayment_rate_by_code.head(5))
```

```
                     violation_description  fraction_paid  \
0  EXPIRED PLATES OR TEMPORARY REGISTRATION       0.604361
1                           STREET CLEANING       0.811612
2              RESIDENTIAL PERMIT PARKING        0.742262
3  EXP. METER NON-CENTRAL BUSINESS DISTRICT       0.792913
4       PARKING/STANDING PROHIBITED ANYTIME       0.705817

   average_level1_fine  n_tickets
0            54.968869      44811
1            54.004249      28712
2            66.338302      23683
3            46.598058      20600
4            66.142864      19753
```

2. Make a scatter plot which shows the relationship between fine amount and the fraction of tickets that are paid. Focus only on violations that appear at least 100 times. Then make two other plots which show the same relationship using different encodings. If there are outliers, you can choose to exclude them from the plot (keeping in mind how this might change the interpretation of the results). For all three plots, write out what are the headlines and at least one sub-message.

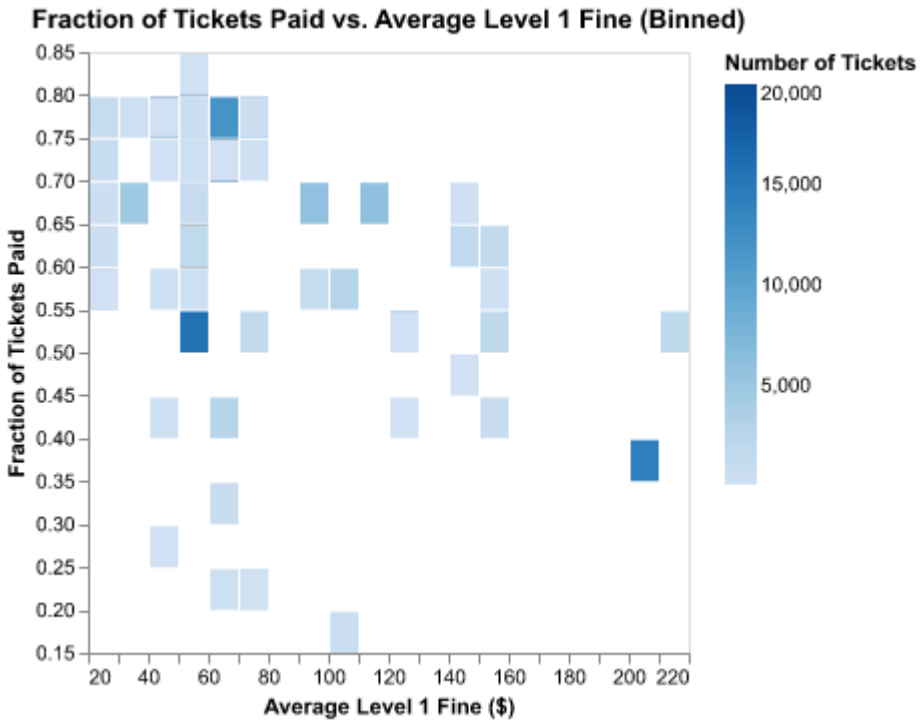
```
# filter to only include violation descriptions that appear at least 100
 ↪  times
filtered_result = repayment_rate_by_code[repayment_rate_by_code['n_tickets']
 ↪  >= 100]

# Create a scatter plot of the fraction of tickets paid versus the fine level
scatter_plot = alt.Chart(filtered_result).mark_circle(size=60).encode(
    x=alt.X('average_level1_fine', title='Average Level 1 Fine ($)'),
    y=alt.Y('fraction_paid', title='Fraction of Tickets Paid')
).properties(
    title='Fraction of Tickets Paid vs. Average Level 1 Fine'
)

scatter_plot
```



- Headline: there is negative relationship between average fine level and fraction paid.
- Submessages:
  - Almost all of the fine values are under $250.
  - There is an outlier at fine value $500.
  - Across violation types there is a lot of variation in the fraction of tickets paid, ranging from 20% to almost 85%.

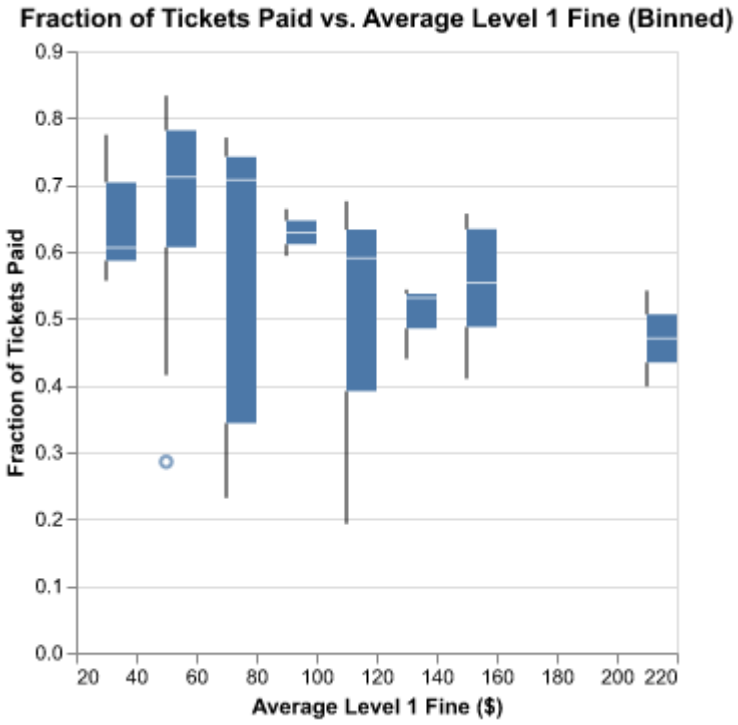
```
# drop the outlier with fine over 400
filtered_result = filtered_result[filtered_result['average_level1_fine'] <
 ↪  400]

binned_scatter_plot = alt.Chart(filtered_result).mark_bar().encode(
    alt.X('average_level1_fine:Q', title='Average Level 1 Fine ($)',
 ↪  bin=alt.Bin(maxbins=20)),
    alt.Y('fraction_paid:Q', title='Fraction of Tickets Paid',
 ↪  bin=alt.Bin(maxbins=20)),
    alt.Color('n_tickets:Q', title='Number of Tickets',
        scale=alt.Scale(domain=[100, 20000]))
).properties(
    title='Fraction of Tickets Paid vs. Average Level 1 Fine (Binned)'
)
```

```
binned_scatter_plot
```



- Headline: there is a negative relationship between average fine level and fraction paid.
- Submessages:
  - There seems to be little relationship between the average fine level and the frequency of tickets
  - Similarly, there is little relationship between the fraction of tickets paid and frequency.

```
# Create a scatter plot of the fraction of tickets paid versus the fine level
 ↪  with a regression line
binned_boxplot = alt.Chart(filtered_result).mark_boxplot().encode(
    x=alt.X('average_level1_fine', title='Average Level 1 Fine ($)',
 ↪  bin=alt.Bin(maxbins=10)),
    y=alt.Y('fraction_paid', title='Fraction of Tickets Paid')
).properties(
    title='Fraction of Tickets Paid vs. Average Level 1 Fine (Binned)'
)

binned_boxplot
```



- Headline: the greatest variation in fraction of tickets paid occurs in the middle of the fine level distribution (between $60-120)
- Submessages:
  - There is a negative relationship between fine level and fraction paid.
  - Within bins of fine level, the distribution of fraction of tickets paid is skewed (as indicated by the median not being in the middle of the IQR for many of the box plots)

3. The City Clerk doesn't understand regressions and only has time to look at one plot. Which plot are you going to bring to them and why?

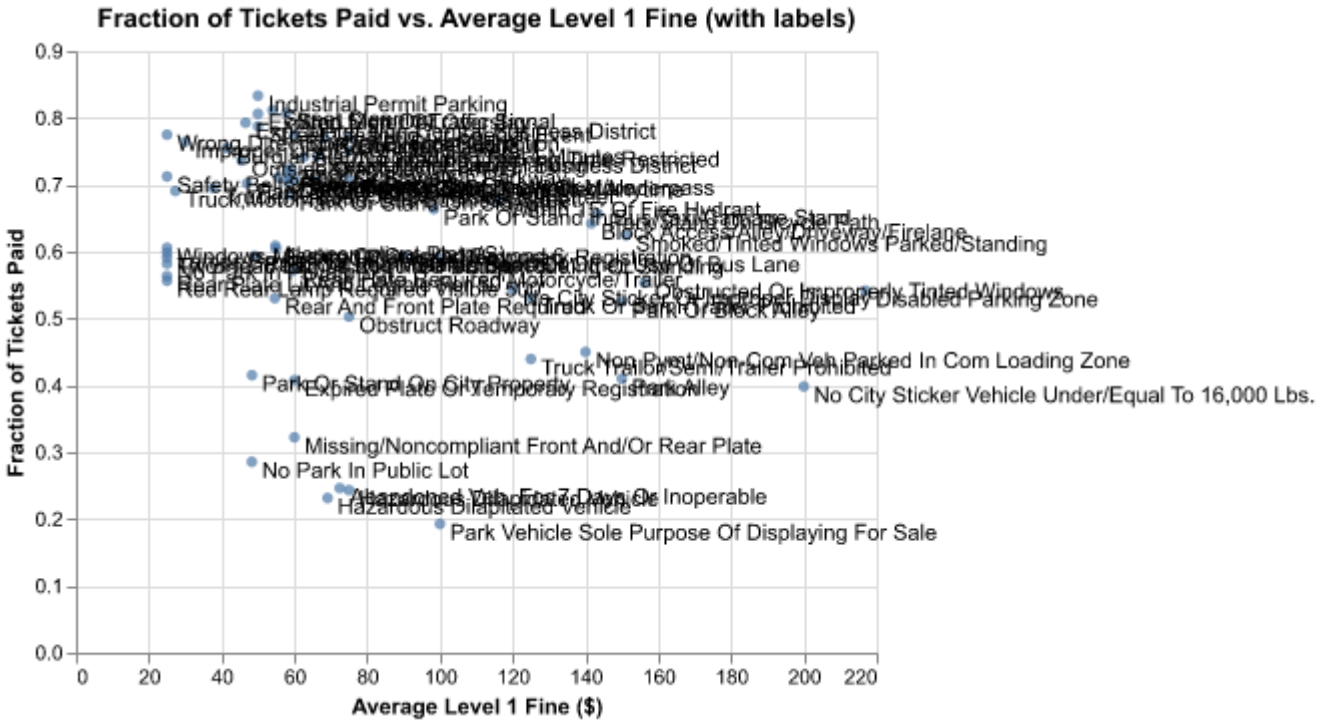
The raw scatter plot does the best job of showing the loose negative relationship between fine size and repayment rate. Keeping the outlier in does not distort the message too much as that point approximately follows the linear relationship in the rest of the data. Since there is not much correlation between ticket frequency and fine size/repayment rate, there is not much useful information to be gleaned from the binned scatter plot. The box plot is best at showing the variation in fraction of tickets paid, but misses the headline message of the negative relationship between fine size and repayment rate. *Note that there are many possible answers to this, and you will receive full-credit for any well thought-out answer.*

Exploration vs. Production (25 %)

1. Go back to your scatter plot on level 1 fine and fraction paid from the previous section. We want to add labels to each dot. Implement this in two ways:

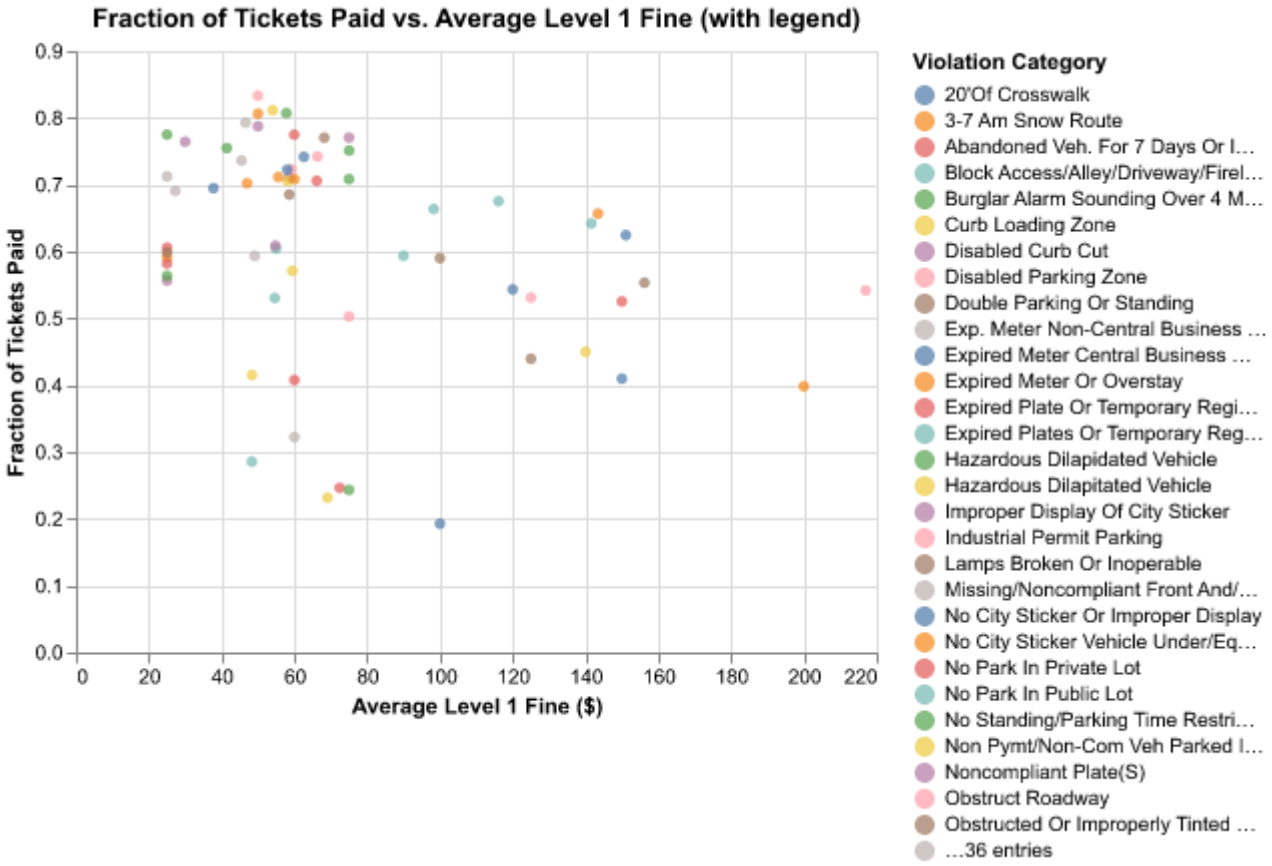
    a. Label every dot with adjacent text or
    b. Put the text in a legend.

```
# make the titles lower case
filtered_result['violation_description'] =
 ↪  filtered_result['violation_description'].str.title()

# (a) Scatter plot with adjacent text labels for all dots
plot_a = alt.Chart(filtered_result).mark_circle().encode(
    x=alt.X('average_level1_fine:Q', title='Average Level 1 Fine ($)'),
    y=alt.Y('fraction_paid:Q', title='Fraction of Tickets Paid')
)

text_a = alt.Chart(filtered_result).mark_text(align='left', dx=5,
 ↪  dy=5).encode(
    x='average_level1_fine:Q',
    y='fraction_paid:Q',
    text='violation_description:N'
)

chart_a = (plot_a + text_a).properties(
    width=400,
    height=300,
    title='Fraction of Tickets Paid vs. Average Level 1 Fine (with labels)'
)

# chart_a.save('figures/chart_a.png', scale_factor=3.0)
chart_a
```



This is unreadable!

```
# (b) Scatter plot with legend for all dots
chart_b = alt.Chart(filtered_result).mark_circle().encode(
    x=alt.X('average_level1_fine:Q', title='Average Level 1 Fine ($)'),
    y=alt.Y('fraction_paid:Q', title='Fraction of Tickets Paid'),
    color=alt.Color('violation_description:N',
 ↪  legend=alt.Legend(title='Violation Category'))
).properties(
    width=400,
    height=300,
    title='Fraction of Tickets Paid vs. Average Level 1 Fine (with legend)'
)

chart_b
```



Better, but still not useful!

2. Either way, you will find the same problem – there are too many labels and the plot is illegible.

    a. First, write a proposal for improvement in words.
    b. Implement your proposal for the improvement in code and make a new plot.

My proposal is to pick the ten most commonly used violation descriptions and the rest of the dots be labeled as "Other". This will make the plot more readable by reducing the number of labels and making the plot more focused on the ten most common violation types.

```
# 1. Easy way: Pick the ten most commonly used violation descriptions
top_10_violations = filtered_result.nlargest(10,
 ↪  'n_tickets')['violation_description'].tolist()
filtered_result['category'] = 'Other'
filtered_result.loc[filtered_result['violation_description'].isin(top_10_violations),
 ↪  'category'] = filtered_result['violation_description']
```

```
chart_top_10 = alt.Chart(filtered_result).mark_circle().encode(
    x=alt.X('average_level1_fine:Q', title='Average Level 1 Fine ($)'),
    y=alt.Y('fraction_paid:Q', title='Fraction of Tickets Paid'),
    color=alt.Color('category:N', scale=alt.Scale(scheme='category20'))
).properties(
    width=400,
    height=300,
    title='Top 10 Violations and Others'
)

chart_top_10
```