Problem Set 1

Jennifer Edouard

October 6, 2024

*1. “This submission is my work alone and complies with the 30538 integrity policy.” Add*

*your initials to indicate your agreement: \*\*JE\*\**

*2. “I have uploaded the names of anyone I worked with on the problem set here” \*\*JE\*\**

*(1 point)*

*3. Late coins used this pset: \*\*1\*\* Late coins left after submission: \*\*3\*\**

import pandas as pd

import altair as alt

## **1. Read in one percent sample (15 points)**

**1.1** To help you get started, we pushed a file to the course repo called parking\_tickets\_one\_percent.csv which gives you a one percent sample of tickets. We constructed the sample by selecting 1 ticket numbers that end in 01. How long does it take to read in this file? (Find a function to measure how long it takes the command to run. Note that everytime you run, there will be some difference in how long the code takes to run). Add an assert statement which verifies that there are 287458 rows.

*Sources: https://www.geeksforgeeks.org/how-to-check-the-execution-time-of-python-script/*

import time

start = time.time()  
  
base\_path = r"C:\Users\jenni\OneDrive - The University of Chicago\2-Python II\Github\ppha30538\_fall2024\problem\_sets\ps1\data\parking\_tickets\_one\_percent.csv"  
  
df\_ps1 = pd.read\_csv(base\_path)  
  
end = time.time()  
  
print("It took", end - start, "seconds to read in the data file")

It took 0.9196999073028564 seconds to read in the data file

C:\Users\jenni\AppData\Local\Temp\ipykernel\_38216\3909265210.py:5: DtypeWarning: Columns (7) have mixed types. Specify dtype option on import or set low\_memory=False.  
 df\_ps1 = pd.read\_csv(base\_path)

assert len(df\_ps1) == 287458

**1.2** Using a function in the os library calculate how many megabytes is the CSV file? Using math, how large would you predict the full data set is?

*Sources: https://www.geeksforgeeks.org/get-file-size-in-bytes-kb-mb-and-gb-using-python/*

import os

file\_info = os.stat(base\_path)  
file\_size\_bytes = file\_info.st\_size  
  
file\_size\_mb = file\_size\_bytes / (1024 \* 1024)  
  
print(f"The data set is about", file\_size\_mb, "megabytes")

The data set is about 80.05409908294678 megabytes

Knowing that the data is only from ticket numbers ending in 01 and that there are 100 possible combinations of two digits, we can estimate that this data is merely 1% of the entire dataset. I’ll multiply the size of this file by 100 to get a sense of the larger dataset

print(f"The larger data set is about", file\_size\_mb \* 100, "megabytes")

The larger data set is about 8005.409908294678 megabytes

**1.3** The rows on the dataset are ordered or sorted by a certain column by default. Which column? Then, subset the dataset to the first 500 rows and write a function that tests if the column is ordered.

*Sources: https://www.datacamp.com/tutorial/functions-python-tutorial*

The rows seem to be in order of the issue\_date column, which is the column stating the date of the ticket issuance

first\_500\_subset = df\_ps1.head(500)

def ordered(data):  
 return data.is\_monotonic\_increasing  
  
check\_yes = ordered(first\_500\_subset["issue\_date"])  
  
if check\_yes:  
 print("The ticket issuance date column is ordered.")  
else:   
 print("The ticket issuance date column is not ordered")

The ticket issuance date column is ordered.

## **2. Cleaning the data and benchmarking (15 points)**

**2.1** How many tickets were issued in the data in 2017? How many tickets does that imply were issued in the full data in 2017? How many tickets are issued each year according to the ProPublica article? Do you think that there is a meaningful difference?

*Sources: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#boolean-indexing*

only\_2017\_subset = df\_ps1[df\_ps1["issue\_date"].str.startswith("2017")]  
  
print(f"According to our data inclduing only tickets ending in '01', in the year 2017, {len(only\_2017\_subset)} tickets were issued. However, this means that the estimate for all the tickets issued in 2017 is {len(only\_2017\_subset) \* 100}. According to the ProPulica article, 'EACH YEAR, the City of Chicago issues more than 3 million tickets.' Based on that estimate, I would say there is a meaningful difference in our estimate compared to the ProPublica information.")

According to our data inclduing only tickets ending in '01', in the year 2017, 22364 tickets were issued. However, this means that the estimate for all the tickets issued in 2017 is 2236400. According to the ProPulica article, 'EACH YEAR, the City of Chicago issues more than 3 million tickets.' Based on that estimate, I would say there is a meaningful difference in our estimate compared to the ProPublica information.

**2.2** Pooling the data across all years what are the top 20 most frequent violation types?Make a bar graph to show the frequency of these ticket types. Format the graph such that the violation descriptions are legible and no words are cut off.

*Sources: https://stackoverflow.com/questions/53983072/arrange-bar-chart-in-ascending-descending-order & https://vega.github.io/vega/tutorials/bar-chart/*

pip install vega\_datasets

import vega\_datasets

group\_violation = df\_ps1.groupby("violation\_description").size().reset\_index(name = "count")  
group\_violation = group\_violation.sort\_values(by = "count", ascending = False)  
  
top\_20\_violation = group\_violation.head(20)

chart\_1 = alt.Chart(top\_20\_violation).mark\_bar().encode(  
 alt.X("violation\_description", title = "Traffic Violation Description", sort = alt.EncodingSortField(field = "count", order = "descending"),  
 axis = alt.Axis(labelAngle = -30)  
 ),  
 alt.Y("count", title = "Frequency")  
).properties(  
 title = "Top 20 Most Frequently Ticketed Traffic Violations in Chicago, IL",  
 width = 800  
)  
chart\_1

A graph of a graph showing a number of traffic

Description automatically generated with medium confidence

## **3. Visual Encoding (15 points)**

**3.1** In lecture 2, we discussed how Altair thinks about categorizing data series into four different types. Which data type or types would you associate with each column in the data frame? Your response should take the form of a markdown table where each row corresponds to one of the variables in the parking tickets dataset, the first column is the variable name and the second column is the variable type or types. If you argue that a column might be associated with than one type, explain why in writing below the table.

*Sources: https://stackoverflow.com/questions/67997825/python-altair-generate-a-table-on-selection*

| Variable Name | Variable Type(s) |
| --- | --- |
| ticket\_number | Ordinal |
| issue\_date | Temporal |
| violation\_location | Nominal |
| license\_plate\_number | Nominal |
| license\_plate\_state | Nominal |
| license\_plate\_type | Nominal |
| zipcode | Nominal |
| violation\_code | Nominal |
| violation\_description | Nominal |
| unit | Nominal |
| unit\_description | Nominal |
| vehicle\_make | Nominal |
| fine\_level1\_amount | Quantitative |
| fine\_level2\_amount | Quantitative |
| current\_amount\_due | Quantitative |
| total\_payments | Quantitative |
| ticket\_queue | Ordinal |
| ticket\_queue\_date | Temporal |
| notice\_level | Nominal |
| hearing\_disposition | Nominal |
| notice\_number | Ordinal |
| officer | Nominal |
| address | Nominal |

**3.2** Compute the fraction of time that tickets issued to each vehicle make are marked as paid. Show the results as a bar graph. Why do you think that some vehicle makes are more or less likely to have paid tickets?

group\_vehicle\_make = df\_ps1[df\_ps1["current\_amount\_due"] == "0"]  
  
group\_vehicle\_make = df\_ps1.groupby("vehicle\_make").size().reset\_index(name = "count")  
  
group\_vehicle\_make["fraction\_of\_total"] = group\_vehicle\_make["count"] / group\_vehicle\_make["count"].sum()

chart\_2 = alt.Chart(group\_vehicle\_make).mark\_bar().encode(  
 alt.X("vehicle\_make", title = "Vehicle Make", sort = alt.EncodingSortField(field = "count", order = "descending")),  
 alt.Y("fraction\_of\_total", title = "Fraction of Total Tickets")  
).properties(  
 title = "Most Frequently Ticketed Vehicle Makes in Chicago, IL"  
)  
chart\_2

A graph with a bar graph

Description automatically generated with medium confidence

I think that car models that are more likely to belong to higher income people will be paid more promptly than those with low-income. Thus, cars that are more likely to belong to low-income individuals are more likely to have an outstanding balance

**3.3** Make a plot for the number of tickets issued over time by adapting the Filled Step Chart example online

df\_ps1["issue\_date"] = pd.to\_datetime(df\_ps1["issue\_date"])

group\_issue\_date = df\_ps1.groupby(df\_ps1["issue\_date"].astype(str).str[:10]).size().reset\_index(name = "count")

import altair as alt  
  
alt.data\_transformers.disable\_max\_rows()  
  
chart\_3 = alt.Chart(group\_issue\_date).mark\_area(  
 color="hotpink",  
 interpolate='step-after',  
 line=True  
).encode(  
 alt.X("issue\_date:T", title = "Date"),  
 alt.Y("count:Q", title = "Ticket Count")  
).properties(  
 title = "Most Frequently Ticketed Vehicle Makes in Chicago, IL"  
)  
chart\_3

A graph of a ticketed vehicle

Description automatically generated

**3.4** Make a plot for the number of tickets issued by month and day by adapting the Annual Weather Heatmap example online.

df\_ps1["month"] = df\_ps1["issue\_date"].dt.month  
df\_ps1["day"] = df\_ps1["issue\_date"].dt.day  
  
group\_month\_and\_day = df\_ps1.groupby(["month","day"]).size().reset\_index(name = "count")

chart\_4 = alt.Chart(group\_month\_and\_day).mark\_rect().encode(  
 alt.X("day:O").title("Day"),  
 alt.Y("month:O").title("Month"),  
 alt.Color("count:Q",title = "Number of Tickets", scale = alt.Scale(scheme = "blues")),  
 tooltip=[  
 alt.Tooltip("month:O", title="Month"),  
 alt.Tooltip("day:O", title = "Day"),  
 alt.Tooltip("count:Q", title="Number of Tickets"),  
 ]  
).properties(  
 title = "Daily Traffic Violation Tickets Issued in Chicago, IL"  
).configure\_view(  
 step=13,  
 strokeWidth=0  
).configure\_axis(  
 domain=False  
)  
chart\_4

A blue and white pixelated image

Description automatically generated with medium confidence

**3.5** Subset to the five most common types of violations. Make a plot for the number of tickets issued over time by adapting the Lasagna Plot example online.

*Sources: https://stackoverflow.com/questions/45639408/broken-axis-in-altair-vega*

violation\_count\_top\_5 = df\_ps1["violation\_description"].value\_counts().nlargest(5)  
  
top\_5\_violation\_converted = violation\_count\_top\_5.index.tolist()  
  
df\_top\_5\_violations = df\_ps1[df\_ps1['violation\_description'].isin(top\_5\_violation\_converted)]

df\_top\_5\_violations.loc[:, "date\_only\_no\_time"] = df\_top\_5\_violations["issue\_date"].astype(str).str[:10]  
  
top\_5\_day\_and\_violation = df\_top\_5\_violations.groupby(["date\_only\_no\_time", "violation\_description"]).size().reset\_index(name = "count")  
  
top\_5\_day\_and\_violation['date\_only\_no\_time'] = top\_5\_day\_and\_violation['date\_only\_no\_time'].astype(str)

chart\_5 = alt.Chart(top\_5\_day\_and\_violation, width = 500, height = 500).mark\_rect().encode(  
 x = alt.X("date\_only\_no\_time:O", title = "Time",  
 axis = alt.Axis(labelAngle = 0, labelOverlap = "greedy")),  
 y = alt.Y("violation\_description:N", title = "Violation Description",  
 axis = alt.Axis( labelAngle = 0, labelOverlap = "greedy")),  
 color = alt.Color("count:Q", title = "Ticket Count"),  
 tooltip = [  
 alt.Tooltip("date\_only\_no\_time:O", title = "Time"),  
 alt.Tooltip("violation\_description:N", title = "Violation Description"),  
 alt.Tooltip("count:Q", title = "Ticket Count")  
 ]  
).properties(  
 title = "Top 5 Traffic Violations in Chicago,IL over time"  
 )  
chart\_5

A screenshot of a graph

Description automatically generated

**3.6**

Filled Step chart: I am able to see the differences over time. However, the details are so granular that I cannot zero in on any specific part of the data

Heatmap: It’s interesting to see where there are surges in the data, but there were not significant changes per day of the month, so the chart isn’t very helpful

Lasagna Plot: Again, it’s nice to see the most popular violations against one another over time, but there is not such severe variation that would allow me to narrow in on any unique piece of the data

**3.7** I think the step chart is best because you can most clearly see variation over time.