NYPD Civilian Complaints

This project contains data on 12,000 civilian complaints filed against New York City police officers. Interesting questions to consider include:

- Does the length that the complaint is open depend on ethnicity/age/gender?
- Are white-officer vs non-white complaintant cases more likely to go against the complainant?
- Are allegations more severe for cases in which the officer and complaintant are not the same ethnicity?
- Are the complaints of women more succesful than men (for the same allegations?)

There are a lot of questions that can be asked from this data, so be creative! You are not limited to the sample questions above.

Getting the Data

The data and its corresponding data dictionary is downloadable [here (https://www.propublica.org/datastore/dataset/civilian-complaints-against-new-york-city-police-officers)](https://www.propublica.org/datastore/dataset/civilian-complaints-against-new-york-city-police-officers). The data dictionary is in the project03 folder.

Note: you don't need to provide any information to obtain the data. Just agree to the terms of use and click "submit."

Cleaning and EDA

- Clean the data.
  - Certain fields have "missing" data that isn't labeled as missing. For example, there are fields with the value "Unknown." Do some exploration to find those values and convert them to null values.
  - You may also want to combine the date columns to create a `datetime` column for time-series exploration.
- Understand the data in ways relevant to your question using univariate and bivariate analysis of the data as well as aggregations.

Assessment of Missingness

- Assess the missingness per the requirements in `project03.ipynb`

Hypothesis Test / Permutation Test

Find a hypothesis test or permutation test to perform. You can use the questions at the top of the notebook for inspiration.

Summary of Findings

Introduction

The dataset implemented in this project is the data of civilian complaining against New York City police officers from *New York City's Civilian Complaint Review Board*. Crucial information such as the ethnicity of the police officer and the rank of him/her when the incident happened are recorded, which are useful to our investigation of whether there exists certain association between a police officer's ethnicity and his/her rank during the incident.

Cleaning and EDA

For data cleaning process, we converted the year and month the case received and closed to more direct datetime columns `time_received` and `time_closed` . Then, we counted all the values and their numbers in every single columns, and found that values such as "Unknown", "Not described", or "Refused" should be counted as NaN, values smaller than 0 should be viewed as NaN as well since it cannot be negative.

To have a better understanding toward the columns of data we interested in, we plot distribution graphs and groupby tables to visualize these columns. The distribution graphs tell us the spread and the weight of each value in different categories, and the groupby tables give us babsic ideas about the missingness association between the data we grouped by and the rest of the columns.

For numerical data, we also introduced box plot to have a better understanding of the center, the range where most of the data falls in, and the outliers.

Based on the data, we found that `complainant_ethnicity` , `mos_ethnicity` and `rank_abbrev_incident` seems to have some kind of association in missingness according the the groupby tables.

Assessment of Missingness

We started off by finding all the columns that contain missing values. Among these columns, we decided to select `complainant_ethnicity` as the column for missingness assessment, and this column's missingness is assessed through the likelihood of its dependence on `mos_ethnicity` and `rank_abbrev_incident` , the two key features of our investigation. Before implementing algorithms, we reached our assumption that missingness in `complainant_ethnicity` is not NMAR. Before cleaning, many of the missing data are displayed as "Unknown" or "Not described", which is impossible to relate to the police officer's rank or ethnicity, thus the missing does not depend on the value itself.

We chose a significance level of 0.05, as it is the most common level for most of the data analysis.

As categorical type data, we used total variance distance to assess the missingness between `complainant_ethnicity` and `rank_abbrev_incident` , and got a p-value of 0.647. This p-value suggests that missingness in complainant's ethnicity is not-at-all dependent on the rank of the officer. On the other hand, the missingness assessment between `complainant_ethnicity` and `mos_ethnicity` produced a p-value of 0, which suggests that the missingness in complainant's ethnicity is dependent on the officer's ethnicity, thus it is an MAR missingness via `mos_ethnicity` .

Hypothesis Test

Our hypothesis test information are as follows:

- Null Hypothesis: the ethnicity of the police officer is independent of the police officers' rank.
- Alternative Hypothesis: the ethnicity of the police officer is not independent of the police officers' rank.
- test statistics: Since the variables we are testing are both categorical variables, we used total variance distance for this hypothesis test.
- A significance level of 0.05 as the most common significance level is maintained during this part of investigation
- Conclusion: we reject our null hypothesis with a p-value of 0. The ethnicity of the police officer is not independent of the police officers' rank.

Code

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
```

Cleaning and EDA

Read data from the file path, then clean the data by combining the year and month column to form a datetime column, and replace "Unknown", 'Refused', 'Not described', value small than 0 with NaN.

In [28]:

```
#read csv
df = pd.read_csv('nypd.csv')
cleaned = df.copy()
cleaned["time_received"] = (pd.to_datetime(cleaned['year_received'].astype(
cleaned["time_closed"] = (pd.to_datetime(cleaned['year_closed'].astype(str)
cleaned = cleaned.drop(['year_received', 'month_received', 'year_closed', '
cleaned = cleaned.replace(["Unknown", 'Refused', 'Not described'], np.NaN)
cleaned = cleaned.replace(-1, np.NaN)
cleaned["complainant_age_incident"] = cleaned["complainant_age_incident"].a
cleaned.head()
```

Out[28]:

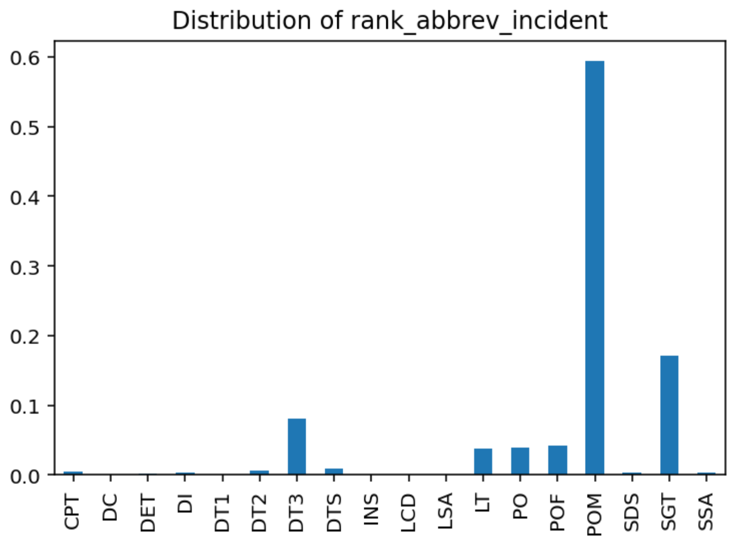
	unique_mos_id	first_name	last_name	command_now	shield_no	complaint_id	command_at_inci
0	10004	Jonathan	Ruiz	078 PCT	8409	42835	078
1	10007	John	Sears	078 PCT	5952	24601	P
2	10007	John	Sears	078 PCT	5952	24601	P
3	10007	John	Sears	078 PCT	5952	26146	P
4	10009	Noemi	Sierra	078 PCT	24058	40253	078

5 rows × 25 columns

Distribution graph for "rank_abbrev_incident", "complainant_ethnicity", "complainant_age_incident", "mos_ethnicity".

In [19]:
```
# Distribution of rank_abbrev_incident
(
    cleaned['rank_abbrev_incident']
    .value_counts(normalize=True)
    .sort_index()
    .plot(kind='bar', title='Distribution of rank_abbrev_incident')
)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x20af84d7790>



In [29]:
```
cleaned.groupby("rank_abbrev_incident").count()
```

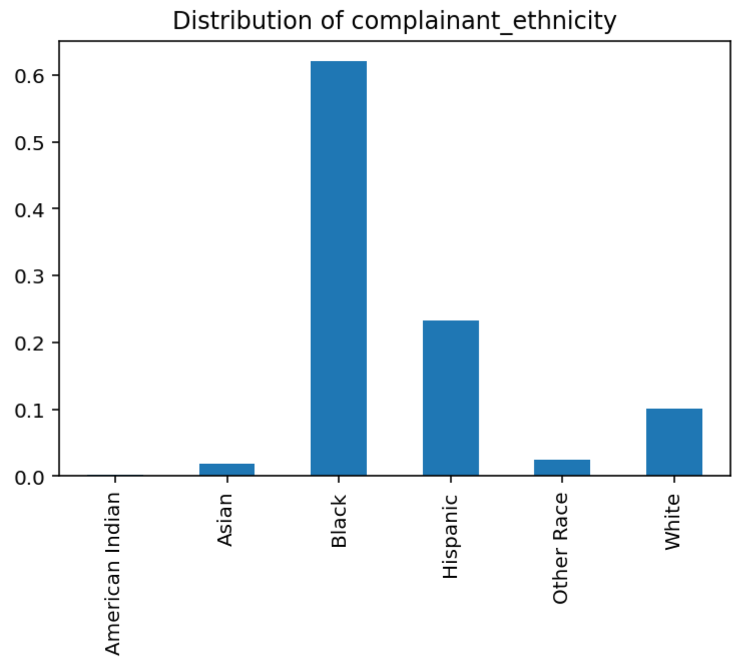
Out[29]:

rank_abbrev_incident	unique_mos_id	first_name	last_name	command_now	shield_no	complaint_id
CPT	182	182	182	182	182	182
DC	2	2	2	2	2	2
DET	50	50	50	50	50	50
DI	96	96	96	96	96	96
DT1	20	20	20	20	20	20
DT2	195	195	195	195	195	195
DT3	2712	2712	2712	2712	2712	2712
DTS	330	330	330	330	330	330
INS	27	27	27	27	27	27
LCD	13	13	13	13	13	13
LSA	24	24	24	24	24	24
LT	1264	1264	1264	1264	1264	1264
PO	1304	1304	1304	1304	1304	1304
POF	1398	1398	1398	1398	1398	1398
POM	19807	19807	19807	19807	19807	19807
SDS	128	128	128	128	128	128
SGT	5701	5701	5701	5701	5701	5701
SSA	105	105	105	105	105	105

18 rows × 24 columns

In [20]:
```
# Distribution of complainant_ethnicity
(
    cleaned['complainant_ethnicity']
    .value_counts(normalize=True)
    .sort_index()
    .plot(kind='bar', title='Distribution of complainant_ethnicity')
)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x20af5ba0700>



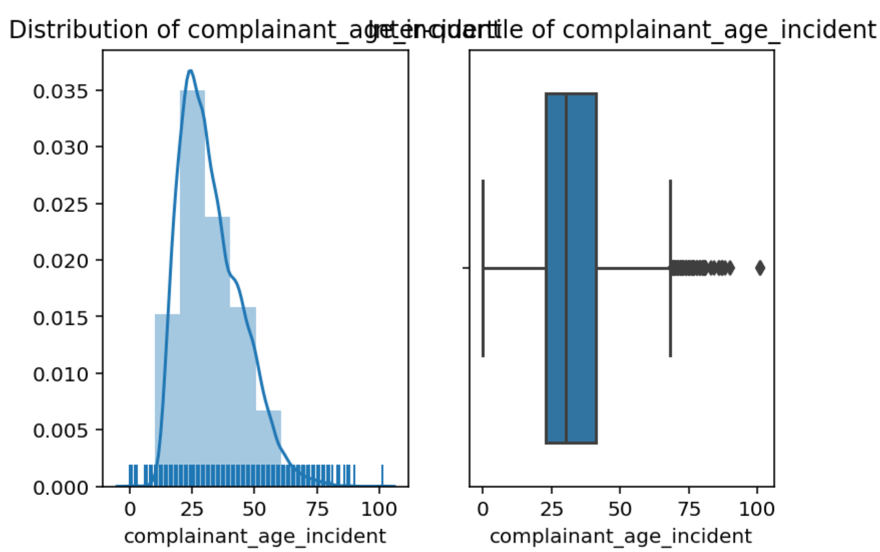
In [30]:
```
cleaned.groupby("complainant_ethnicity").count()
```

Out[30]:

complainant_ethnicity	unique_mos_id	first_name	last_name	command_now	shield_no	complaint_i
American Indian	64	64	64	64	64	6
Asian	532	532	532	532	532	53
Black	17114	17114	17114	17114	17114	1711
Hispanic	6424	6424	6424	6424	6424	642
Other Race	677	677	677	677	677	67
White	2783	2783	2783	2783	2783	278

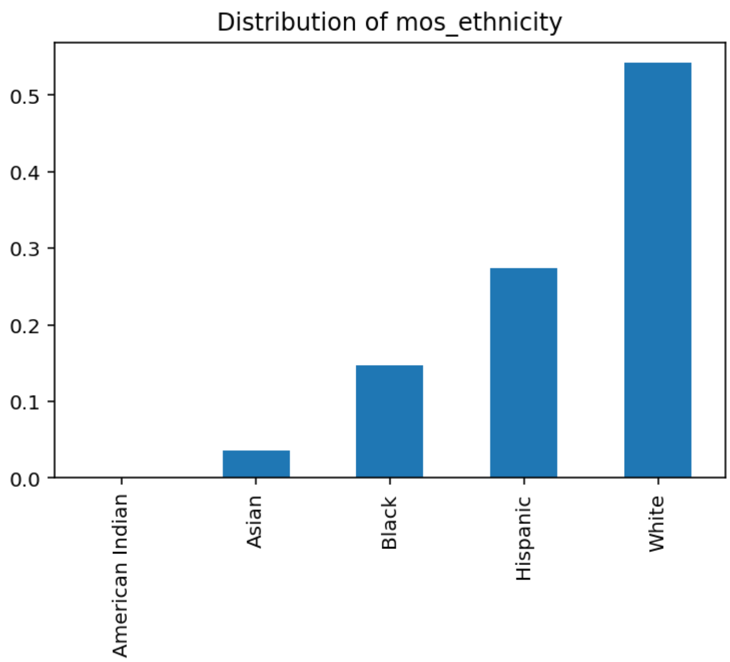
6 rows × 24 columns

In [21]:
```
fig, (dist, box) = plt.subplots(1, 2)

# distribution plot
dist.set_title('Distribution of complainant_age_incident')
sns.distplot(
    cleaned["complainant_age_incident"].values,
    bins=10, hist=True, kde=True, rug=True,
    axlabel='complainant_age_incident',
    ax=dist
)

# box plot
box.set_title('Inter-quartile of complainant_age_incident')
sns.boxplot(cleaned["complainant_age_incident"], ax=box);
```



In [22]:
```
(
    cleaned["mos_ethnicity"]
    .value_counts(normalize=True)
    .sort_index()
    .plot(kind='bar', title='Distribution of mos_ethnicity')
)
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x20af76fb8e0>



In [27]:
```
cleaned.groupby("mos_ethnicity").count()
```

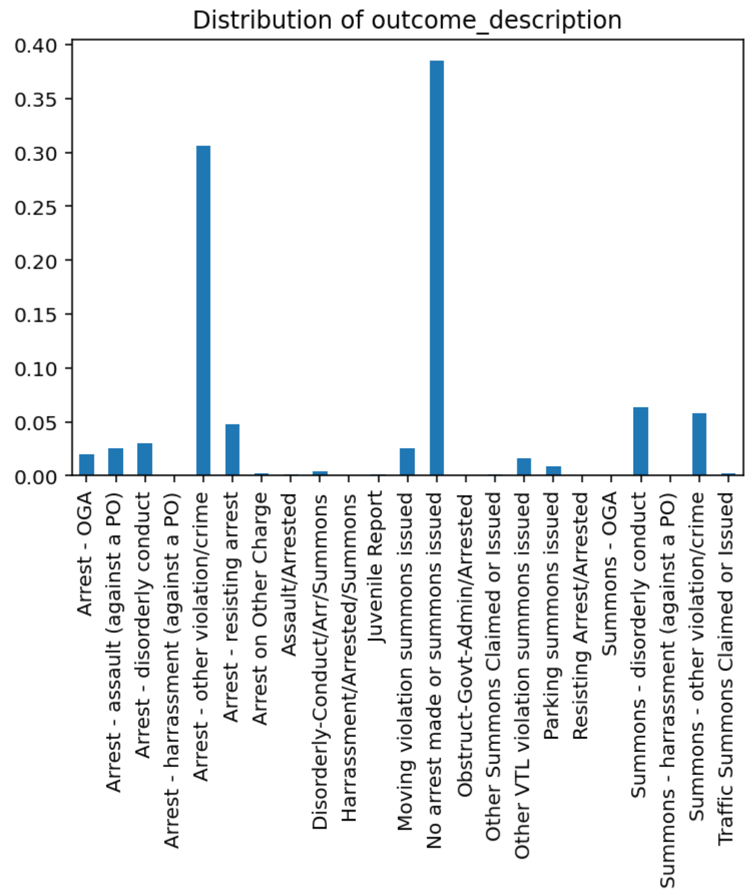
Out[27]:

mos_ethnicity	unique_mos_id	first_name	last_name	command_now	shield_no	complaint_id	comr
American Indian	32	32	32	32	32	32	
Asian	1178	1178	1178	1178	1178	1178	
Black	4924	4924	4924	4924	4924	4924	
Hispanic	9150	9150	9150	9150	9150	9150	
White	18074	18074	18074	18074	18074	18074	

5 rows × 24 columns

In [24]:
```
(
    cleaned["outcome_description"]
    .value_counts(normalize=True)
    .sort_index()
    .plot(kind='bar', title='Distribution of outcome_description')
)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20af77723d0>



In [26]:
```
cleaned.groupby("outcome_description").count()
```

Out[26]:

outcome_description	unique_mos_id	first_name	last_name	command_now	shield_no	c
Arrest - OGA	649	649	649	649	649	
Arrest - assault (against a PO)	852	852	852	852	852	
Arrest - disorderly conduct	1013	1013	1013	1013	1013	
Arrest - harrassment (against a PO)	15	15	15	15	15	
Arrest - other violation/crime	10196	10196	10196	10196	10196	
Arrest - resisting arrest	1593	1593	1593	1593	1593	
Arrest on Other Charge	81	81	81	81	81	
Assault/Arrested	34	34	34	34	34	
Disorderly-Conduct/Arr/Summons	137	137	137	137	137	
Harrassment/Arrested/Summons	8	8	8	8	8	
Juvenile Report	57	57	57	57	57	
Moving violation summons issued	839	839	839	839	839	
No arrest made or summons issued	12822	12822	12822	12822	12822	
Obstruct-Govt-Admin/Arrested	10	10	10	10	10	
Other Summons Claimed or Issued	38	38	38	38	38	
Other VTL violation summons issued	531	531	531	531	531	
Parking summons issued	279	279	279	279	279	
Resisting Arrest/Arrested	25	25	25	25	25	
Summons - OGA	1	1	1	1	1	
Summons - disorderly conduct	2118	2118	2118	2118	2118	
Summons - harrassment (against a PO)	5	5	5	5	5	
Summons - other violation/crime	1940	1940	1940	1940	1940	
Traffic Summons Claimed or Issued	59	59	59	59	59	

23 rows × 24 columns

Assessment of Missingness

To begin with, we started off by finding all the columns with missing values and the proportion of the missing values in the columns

In [246]:
```
null_columns = cleaned.columns[cleaned.isnull().any()]
null_columns = cleaned[null_columns].isnull().sum()/cleaned.shape[0]
null_columns
```

Out[246]:
```
command_at_incident         0.046286
complainant_ethnicity       0.133821
complainant_gender          0.125757
complainant_age_incident    0.144253
allegation                  0.000030
precinct                    0.000719
contact_reason              0.005966
outcome_description         0.001679
dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

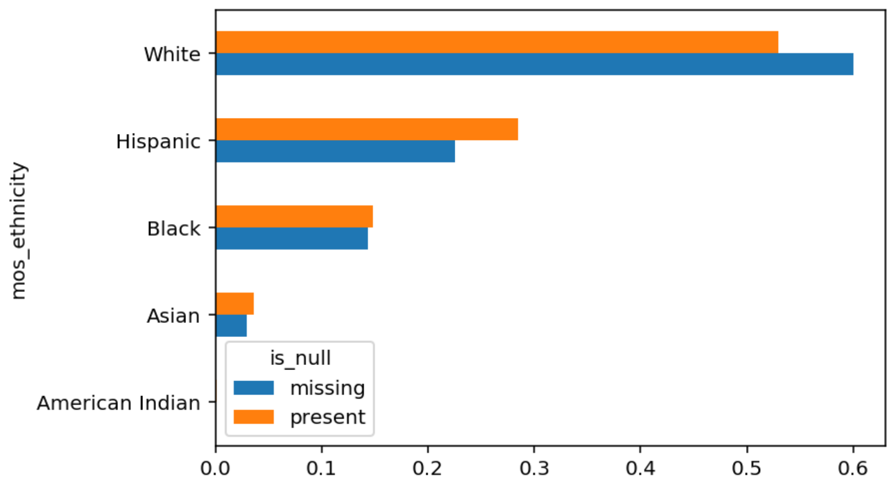
In [250]:
```
#missing keywords: complainant_gender: Not described, ethnicity: [Unknown,
cleaned['complainant_ethnicity'] = cleaned['complainant_ethnicity'].\
replace(['Unknown','Refused'], np.NaN)
cleaned['complainant_gender'] = cleaned['complainant_gender'].replace(['Not
```

In [ ]:

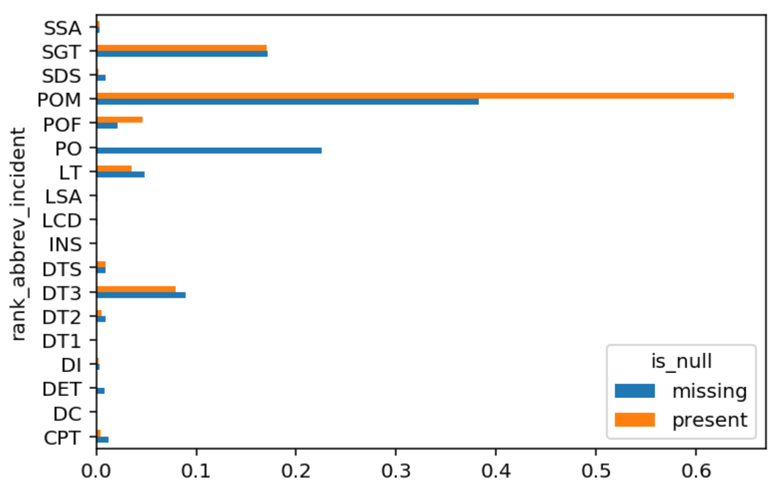
In [ ]:

In [ ]:

In [338]:
```
def nan_plot(data, col1, col2, plot = 'barh'):

    is_null = (
        data[col1]
        .isnull()
        .replace({True: 'missing', False: 'present'})
    )

    distrs = (
        data
        .assign(is_null=is_null)
        .pivot_table(index=col2, columns='is_null', aggfunc='size')
        .apply(lambda x:x/x.sum())
    )

    distrs.plot(kind=plot);
nan_plot(cleaned, 'complainant_ethnicity', 'mos_ethnicity')
```



In [339]:
```
nan_plot(cleaned, 'complainant_ethnicity', 'rank_abbrev_incident')
```



As shown above, bar plots between variables are drown. From the plots, it is obvious that `complainant_ethnicity` and `mos_ethnicity` have a very similar distribution, while `complainant_ethnicity` and `rank_abbrev_incident` have a different distribution.

In [ ]:

We created two extra columns for assessments using permutation tests

In [254]:

```
cleaned['age_null'] = cleaned['complainant_age_incident'].isnull()
cleaned['ethn_null'] = cleaned['complainant_ethnicity'].isnull()
```

The functions for different test statistics, including total variance distance, mean difference, ks-statistics, and sampling and p-value calculations are created.

In [333]:

```
def tvd(data, col, group_col):
    tvd = (
        data
        .pivot_table(
            index=col,
            columns=group_col,
            aggfunc='size',
            fill_value=0
        )
        .apply(lambda x: x / x.sum())
        .diff(axis=1).iloc[:, -1].abs().sum() / 2
        )

    return tvd
def diff_of_means(data, col, groupby):
    data_copy = data.copy()
    data_copy = data_copy.groupby(groupby)[col].mean()
    diff_mean = abs(data_copy.get(key=True) - data_copy.get(key=False))
    return diff_mean
def simulate_null(data, col, groupby, func):

    data_copy = data.copy()
    shuffled = (
        data_copy[col].sample(replace = False, frac=1).reset_index(drop=Tru
    )
    data_copy[col] = shuffled

    return func(data_copy, col, groupby)
def pval(data, col, groupby, func, rep=1000):
    diff = []
    for i in range(rep):
        result = simulate_null(data, col, groupby, func)
        diff.append(result)
    return np.count_nonzero(diff>np.float64(func(data, col, groupby)))/rep


def ks(data, col, groupby):
    from scipy.stats import ks_2samp
    v1 = data[groupby].unique()[0]
    v2 = data[groupby].unique()[1]
    ks_result = ks_2samp(data.loc[data[groupby]==v1, col],data.loc[data[grc
    return ks_result[0]
```

We got a p-value of 0.0 for the missingness in complainant's ethnicity and the officer's ethnicity.

which is less than our significance level, so we determined that the missingness in complainant's ethnicity is dependent of the officer's ethnicity

In [334]:
```
pval(cleaned, 'ethn_null', 'mos_ethnicity',tvd)
```
Out[334]: 0.0

In [ ]:

We got a p-value of 0.647 for the missingness in complainant's ethnicity and the rank of the officer, which is greater than our significance level, so we determined that the missingness in the complainant's ethnicity is independent of the rank of the officer

In [335]:
```
pval(cleaned, 'ethn_null', 'rank_abbrev_incident',tvd)
```
Out[335]: 0.647

In [ ]:

In [ ]:

Hypothesis Test

In [336]:
```
pval(cleaned, 'rank_abbrev_incident', 'mos_ethnicity', tvd) #alternative hy
```
Out[336]: 0.0

The detailed process is written in the finding summary. In conclusion, we reject our null hypothesis with a p-value of 0. The ethnicity of the officer is dependent on the officer's rank.

Discussion & Conclusion

Overall, even though we reached our conclusion that the rank of the officer and the officer's ethnicity have a dependent relationship, which shall satisfy this porject's requirement of performing a hypothesis test. There are more we can do based on our current observation. Since both variables are categorical, the plots we have learned so far may not be the best to demonstrate their relationship. However, as long as we can obtain, we can use a parallel categories diagram to help us visualize and explore further into this relationship.

We have also noticed that columns such as the description and alignment could potentially be considered relatable to our investigation. However, with the knowledge we have right now, a successful analysis on the input string type may not be plausible.

In [ ]: