nypd-2

November 12, 2021

1 NYPD Civilian Complaints

2 Summary of Findings

2.0.1 Introduction

The dataset we chose to investigate in this project contains data about 12,000 civilian complaints against New York City police officers from September 1985 to January 2020. This dataset contains a lot of details about each complaint such as the officers ID, rank at the time of the complaint and present rank, category of alleged misconduct, and a description of the misconduct as well.

We had a look at the suggested questions provided and were drawn to the question concerning whether the complaints of women are more successful than men (for the same allegations). This dataset is perfect to answer this question since it contains all the information we need to come to a conclusion, like the gender of the complainant, allegation, and outcome. We decided that a successful complaint would be when the board_disposition outcome of the complaint is 'Substantiated'.

After having a look at the dataset, we came up with another question we were interested in: Are white-officers more likely to be exonerated or unsubstantiated? We were particularly interested in observing whether white-officers are more likely to get off the hook for a complaint. We simply observing whether there the chance of a white officer getting off the hook is more likely than officers of other ethnicities and not making any conclusions about the causes since we have no proof as to whether the complaint / allegation made is true or false.

2.0.2 Cleaning and EDA

Our initial approach was to first create a datetime column as mentioned in the instructions to find some pattern. We used getting_df function to do so. However, we quickly realised that our questions do not have any relationship with time, so we quickly moved on to clean the data in different way.

To answer the questions we shortlisted, we went to each column of interest and first identified whether there were null values present. We found that there was only one null value in the 'allegation' column and we dropped this row of complaints since it is difficult to classify it if the value is null. We conducted the same checks for the other columns like 'complainant_gender', 'mos_ethnicity' and 'board_disposition' and dropped the rows with null values present.

To further clean the dataset and better answer our questions, we decided to further clean up the 'allegation' column since there were a number of similar values with just a small difference in upper or lower case or slightly different wording. For example, we found that there were a number of allegations called 'Person Searched' and 'Search (of person)' which are the same thing. We designed a helper method called allegation_cleaner to apply to the allegation column and clean it up and better categorise allegations that are similar. After applying this function, the number of unique values in the column went from 115 to 19. Since we are comparing the male and female complaint success rates for each allegation, it is important to have a larger group of complaints which fall under a specific allegation otherwise it can create a flawed method of comparison when conducting the permutation test. We created a similar method to clean up the 'board_disposition' column since there were many unique values which all fell under the category of 'Substantiated'.

To clean the data and answer our first question, we dropped all rows where the gender did not match 'Male' or 'Female' since our question is specifically investigating the difference in complaint success rates between males and females.

2.0.3 Assessment of Missingness

We conducted two permutation tests to assess the missingness of columns in our dataset. We were not able to find any columns which had thier missingness dependent on another column. Moreover, to answer our questions, we only needed 4 columns: complainant_gender, mos_ethnicity, allegation, and board_disposition.

We felt that the complainant_gender was MAR, and the other columns did not have enough missing values to succesfully determine what type of missingness they are.

The first permutation test we conducted was to check whether the missingness of the complainant_gender column depended on board_disposition. We decided to conduct the test with a significance level of 0.05. Our test resulted in a p-value of 0, so we came to the conclusion that the missingness of complainant_gender is not dependent on board_disposition.

The second permutation test we conducted was to check whether the missingness of the complainant_gender column depended on mos_gender (officer's gender). We decided to conduct the test with a significance level of 0.05. Our test resulted in a p-value of 0.83, which is much larger than 0.05, so we came to the conclusion that the missingness of complainant_gender does depend on mos_gender (officer's gender). This could be because male officers might have made females uncomfortable and scared of adding thier gender on the complaint to prevent themselves from being identified.

2.0.4 Hypothesis Test

Question 1: Are the complaints of women more successful than men (for the same allegations)? We consider a complaint as sucessful when the board_disposition is 'Substantiated'.

- Null Hypothesis: Complaints of women have the same success rate as that of men.
- Alternative Hypothesis: Complaints of women are more successful (higher in proportion) than that of men.
- Test Statistic: We are summing the difference in proportions of substantiated complaints between men and women for each allegation

We are conducting the permutation test with a significance level of 0.05.

From the permutation test we conducted, we found a p-value of 0.291, so we fail to reject the null hypothesis and cannot conclude that there is a significant difference in success rates between men and women for each allegation.

Question 2: Are white-officers more likely to be exonerated or unsubstantiated?

- Null Hypothesis: White officers are equally likely to be exonerated or unsubstantiated.
- Alternative Hypothesis: White officers more likely to be exonerated or unsubstantiated.
- Test Statistic: Average difference between white officers exonerated/unsubstantiated and non-white officers exonerated/unsubstantiated

We are conducting the permutation test with a significance level of 0.05.

Our permutation test resulted in a p-value of 0, so we can confidently reject the null hypothesis that white-officers are equally likely to be exonerated or unsubstantiated. This could suggest that the board may be biased towards white-officers when deciding if the allegation is true or not. There could also be some confounding factors that we are not aware of.

3 Code

[1]:
```python
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
```

3.0.1 Cleaning and EDA

[2]:
```python
# Reading the dataset on civilian complaints from csv to pandas Dataframe.
nypd = pd.read_csv('allegations_202007271729.csv')
# Renaming 'unique_mos_id' to 'officer_id' for better readability
nypd.rename(columns={'unique_mos_id': 'officer_id'}, inplace=True)
```

[3]:
```python
# Getting summary of data
nypd.describe()
```

[3]:
```
         officer_id     shield_no  complaint_id  month_received  \
count  33358.000000  33358.000000  33358.000000    33358.000000
mean   18169.912495   6451.608819  23905.058217        6.323551
std     9566.316896   7945.641596  11954.434138        3.362951
min        2.000000      0.000000    517.000000        1.000000
25%     9671.000000   1089.000000  13684.750000        3.000000
50%    19215.000000   3691.000000  25132.000000        6.000000
75%    25412.000000   7155.000000  34252.000000        9.000000
max    36374.000000  31977.000000  43703.000000       12.000000

       year_received  month_closed   year_closed  mos_age_incident  \
count   33358.000000  33358.000000  33358.000000      33358.000000
mean     2010.726782      6.470772   2011.525661         32.346873
```

```
std         6.034725      3.343372      6.085907            6.040944
min      1985.000000      1.000000   1985.000000           20.000000
25%      2007.000000      4.000000   2008.000000           28.000000
50%      2012.000000      6.000000   2013.000000           31.000000
75%      2016.000000      9.000000   2016.000000           36.000000
max      2020.000000     12.000000   2020.000000           60.000000

       complainant_age_incident      precinct
count              28546.000000  33334.000000
mean                  32.484201     64.365213
std                   28.408963     31.451592
min                -4301.000000      0.000000
25%                   23.000000     43.000000
50%                   30.000000     67.000000
75%                   41.000000     81.000000
max                  101.000000   1000.000000
```

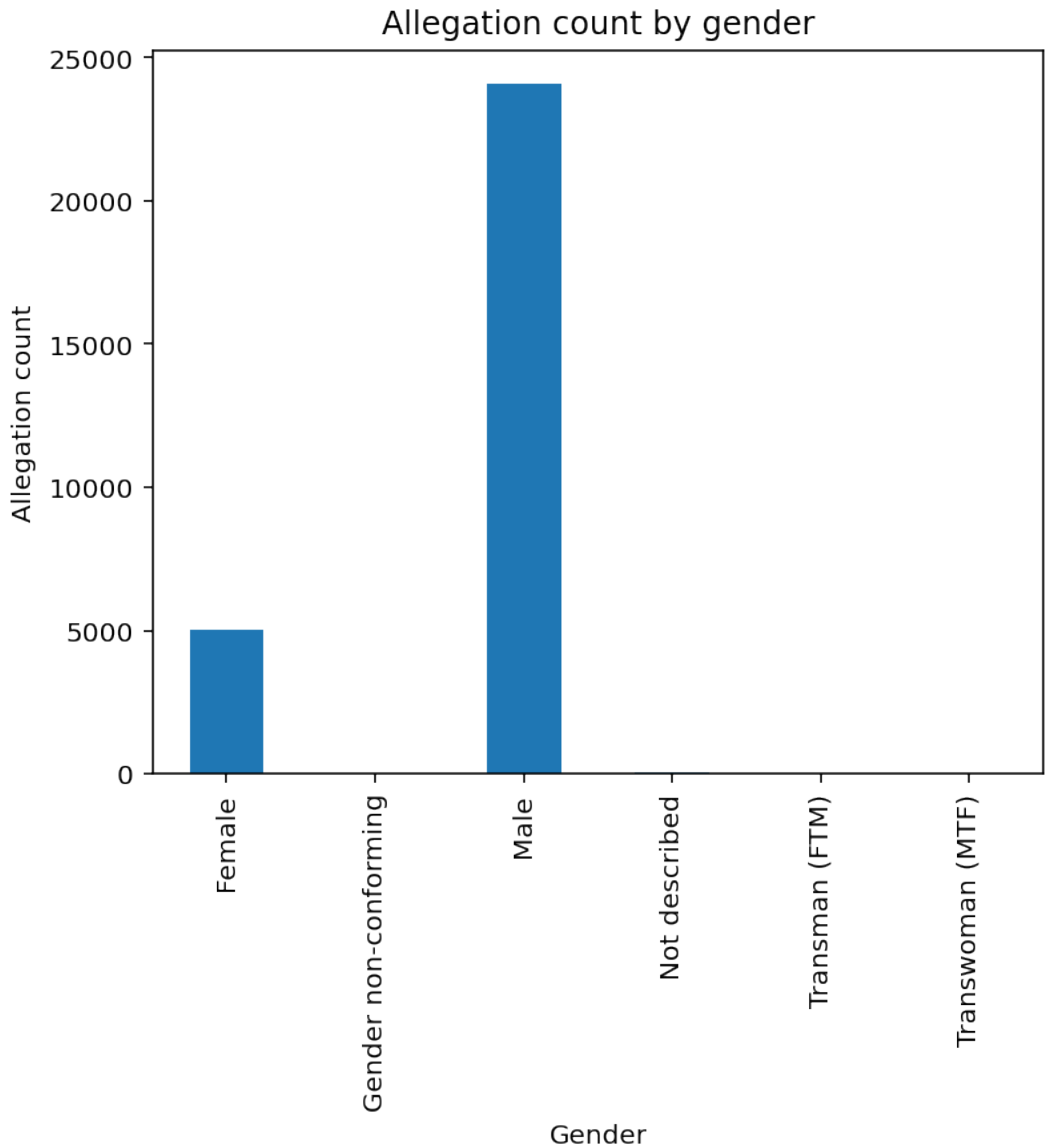
[4]:
```python
# Created function to get the range from month received to month closed
#Can be useful for time-series exploration
def getting_tf(df):
    month = 12
    received = month - df['month_received']
    closed = df['month_closed']
    year_diff = (df['year_closed'] - df['year_received']) * 12
    df['Date Time (Month)'] = year_diff + received + closed
```

[5]:
```python
# Checking how many allegations were made by each gender
nypd.groupby('complainant_gender').count()['allegation']
```

[5]:
```
complainant_gender
Female                    5021
Gender non-conforming        2
Male                     24058
Not described               57
Transman (FTM)               5
Transwoman (MTF)            20
Name: allegation, dtype: int64
```

[6]:
```python
# Plotting the difference in number of allegations by different genders
dfg = nypd.groupby(['complainant_gender'])['allegation'].count()

dfg.plot(kind='bar', title='Allegation count by gender', ylabel='Allegation␣
 ↪count',
         xlabel='Gender', figsize=(6, 5))
```

[6]:
```
<AxesSubplot:title={'center':'Allegation count by gender'}, xlabel='Gender',
ylabel='Allegation count'>
```



[7]:
```
# Checking proportion of male successful complaints
nypd[(nypd['complainant_gender'] == 'Male') & (nypd['board_disposition'] ==␣
↪'Substantiated')].shape[0] / nypd[nypd['complainant_gender'] == 'Male'].
↪shape[0]
```

[7]: 0.0

[8]:
```
# Checking proportion of female successful complaints
nypd[(nypd['complainant_gender'] == 'Female') & (nypd['board_disposition'] ==␣
↪'Substantiated')].shape[0] / nypd[nypd['complainant_gender'] == 'Female'].
↪shape[0]
```

[8]: 0.0

The fact that females have a lower proportion of complaints succesful overall compared to men does not signify anything because our question is asking whether there is a difference in success rates for each allegation.

[9]:
```
# Getting number of nan values in allegation column
nypd['allegation'].isna().value_counts()
```

[9]:
```
False    33357
True         1
Name: allegation, dtype: int64
```

[10]:
```
# Getting number of nan values in complainant_gender column
nypd['complainant_gender'].isna().value_counts()
```

[10]:
```
False    29163
True      4195
Name: complainant_gender, dtype: int64
```

[11]:
```
# Getting number of nan values in board_disposition column
nypd['board_disposition'].isna().value_counts()
```

[11]:
```
False    33358
Name: board_disposition, dtype: int64
```

[12]:
```
# Getting number of nan values in mos_ethnicity column
nypd['mos_ethnicity'].isna().value_counts()
```

[12]:
```
False    33358
Name: mos_ethnicity, dtype: int64
```

Cleaning up the data and removing all rows where data in the columns we are interested in are missing / null:

[13]:
```
# Removing rows where complainant gender is missing (Nan)
# Removing rows where allegation is missing (Nan)
# Removing rows where board_disposition is missing (Nan)
# Removing rows where mos_ethnicity is missing (Nan)

nypd = nypd[nypd['complainant_gender'].notna()]
nypd = nypd[nypd['allegation'].notna()]
nypd = nypd[nypd['board_disposition'].notna()]
nypd = nypd[nypd['mos_ethnicity'].notna()]
```

This is a helper method to clean the allegation colum and reduce the number of unique values to better answer our question (q1). Reduces number of unique values in allegation column from 115 to 19.

[14]:

```python
def allegation_cleaner(x):
    racial_words = ['white', 'black', 'asian', 'ethnic', 'race', 'oriental', 'hispanic', 'mexican', 'indian', 'jewish', 'racial', 'immigration', 'gay', 'gender', 'religion']
    physical_force_words = ['force', 'mace', 'club', 'restricted', 'slap', 'push', 'kick', 'punch', 'shove', 'choke', 'pepper', 'dragged', 'beat', 'tight', 'restrain', 'forcible', 'hit', 'damage']
    if 'sex' in str(x).lower():
        return 'Sexual Misconduct'
    elif any([force_word in str(x).lower() for force_word in physical_force_words]):
        return 'Unnecessary Physical Force'
    elif any([word in str(x).lower() for word in racial_words]):
        return 'Discrimination / Bias'
    elif ('drawn' in str(x).lower()) or ('pointed' in str(x).lower()):
        return 'Gun Pointed'
    elif ('threat of force' in str(x).lower()) or ('threat' in str(x).lower()):
        return 'Threat of Force (verbal or physical)'
    elif ('threat of force' in str(x).lower()):
        return 'Threat of Force (verbal or physical)'
    elif ('threat of force' in str(x).lower()):
        return 'Threat of Force (verbal or physical)'
    elif ('word' in str(x).lower()) or ('curse' in str(x).lower()) or ('abuse' in str(x).lower()) or ('question' in str(x).lower()):
        return 'Verbal Abuse'
    elif ('frisk' in str(x).lower()) or ('search (of person)' in str(x).lower()) or ('person searched' in str(x).lower()):
        return 'Frisk'
    elif ('vehicle' in str(x).lower()) or ('stop' in str(x).lower()):
        return 'Vehicle (Search or Stop)'
    elif ('premise' in str(x).lower()):
        return 'Premises entered and/or searched'
    elif ('fired' in str(x).lower()):
        return 'Gun Fired'
    elif ('demeanor' in str(x).lower()) or ('gesture' in str(x).lower()) or ('action' in str(x).lower()) or ('discourtesy' in str(x).lower()):
        return 'Discourtesy / Rude'
    elif ('refusal' in str(x).lower()) or ('deletion' in str(x).lower()) or ('interference' in str(x).lower()) or ('retaliatory' in str(x).lower()) or ('dissemination' in str(x).lower()) or ('rtka' in str(x).lower()) or ('interpretation' in str(x).lower()):
        return 'Rules for arrest not followed'
    elif ('strip' in str(x).lower()) or ('cavity' in str(x).lower()):
        return 'Strip-searched'
    else:
        return x
```

```
nypd['allegation'] = nypd['allegation'].apply(allegation_cleaner)
```

[15]:
```
# Cleaning board disposition column

def board_disposition_cleaner(x):
    if 'unsubstantiated' in str(x).lower():
        return 'Unubstantiated'
    elif 'exonerated' in str(x).lower():
        return 'Exonerated'
    else:
        return 'Substantiated'

nypd['board_disposition'] = nypd['board_disposition'].
 ↪apply(board_disposition_cleaner)
```

[16]:
```
# Checking number of different board_disposotion outcomes for each gender after␣
 ↪cleaning up board_disposition column

nypd.pivot_table(index='complainant_gender', columns='board_disposition',␣
 ↪aggfunc='size')
```

[16]:

board_disposition complainant_gender	Exonerated	Substantiated	Unubstantiated
Female	1415.0	1032.0	2574.0
Gender non-conforming	NaN	2.0	NaN
Male	6571.0	6151.0	11336.0
Not described	21.0	14.0	22.0
Transman (FTM)	NaN	2.0	3.0
Transwoman (MTF)	4.0	11.0	5.0

Creating new dataframe with complaints only from men and women to help us answer the question regarding male and female complaint success rates.

[17]:
```
# Creating new dataframe with complaints only from men and women
only_m_and_f = nypd[(nypd['complainant_gender'] == 'Male') |␣
 ↪(nypd['complainant_gender'] == 'Female')]
```

3.0.2 Assessment of Missingness

We checked our columns of interest (gender, allegation, ethnicity, and board_disposition for missingness in the Data Cleaning section and dropped all rows with NaN values.

[18]:
```
# Getting sum of NaN values in each column
nypd.isnull().sum(axis = 0)
```

[18]:
```
officer_id                   0
first_name                   0
last_name                    0
command_now                  0
shield_no                    0
complaint_id                 0
month_received               0
year_received                0
month_closed                 0
year_closed                  0
command_at_incident        142
rank_abbrev_incident         0
rank_abbrev_now              0
rank_now                     0
rank_incident                0
mos_ethnicity                0
mos_gender                   0
mos_age_incident             0
complainant_ethnicity      288
complainant_gender           0
complainant_age_incident   680
fado_type                    0
allegation                   0
precinct                    19
contact_reason             140
outcome_description         45
board_disposition            0
dtype: int64
```

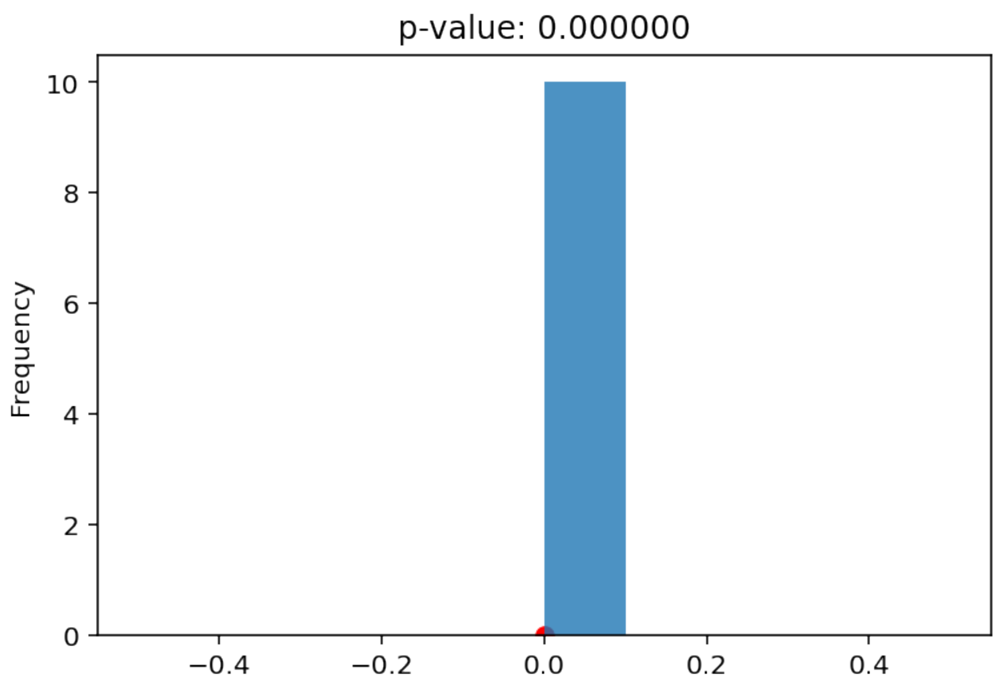
Conducting a permutation test to assess whether the missingness of the gender column is dependent on board_disposition:

[19]:
```
mcar_nypd = nypd[['complainant_gender', 'board_disposition', 'allegation']]
distr = (
    mcar_nypd
    .assign(is_null=mcar_nypd['complainant_gender'].isnull())
    .pivot_table(index='is_null', columns='board_disposition', aggfunc='size')
)
distr = (distr.T / distr.sum(axis=1)).T

n_repetitions = 100

tvds = []
for _ in range(n_repetitions):

    # shuffle the board_disposition column
    shuffled_col = (
```

```
        mcar_nypd['board_disposition']
        .sample(replace=False, frac=1)
        .reset_index(drop=True)
    )

    # put them in a table
    shuffled = (
        mcar_nypd
        .assign(**{
            'board_disposition': shuffled_col,
            'is_null': mcar_nypd['complainant_gender'].isnull()
        })
    )

    # compute the tvd
    shuffled = (
        shuffled
        .pivot_table(index='is_null', columns='board_disposition',␣
↪aggfunc='size')
        .apply(lambda x:x / x.sum(), axis=1)
    )

    tvd = shuffled.diff().iloc[-1].abs().sum() / 2
    # add it to the list of results
    tvds.append(tvd)


obs = distr.diff().iloc[-1].abs().sum() / 2


# Calculating pvalue
pval = np.mean(tvds > obs)
pval
```

[19]: 0.0

[20]:
```
# Plotting graph to visualize p-value

pd.Series(tvds).plot(kind='hist', density=True, alpha=0.8, title='p-value: %f'␣
↪% pval)
plt.scatter(obs, 0, color='red', s=40);
```



Conclusion Since the p value we obtained is 0, we can conclude that the missingness of the complainant_gender column is not dependent on board_disposition.

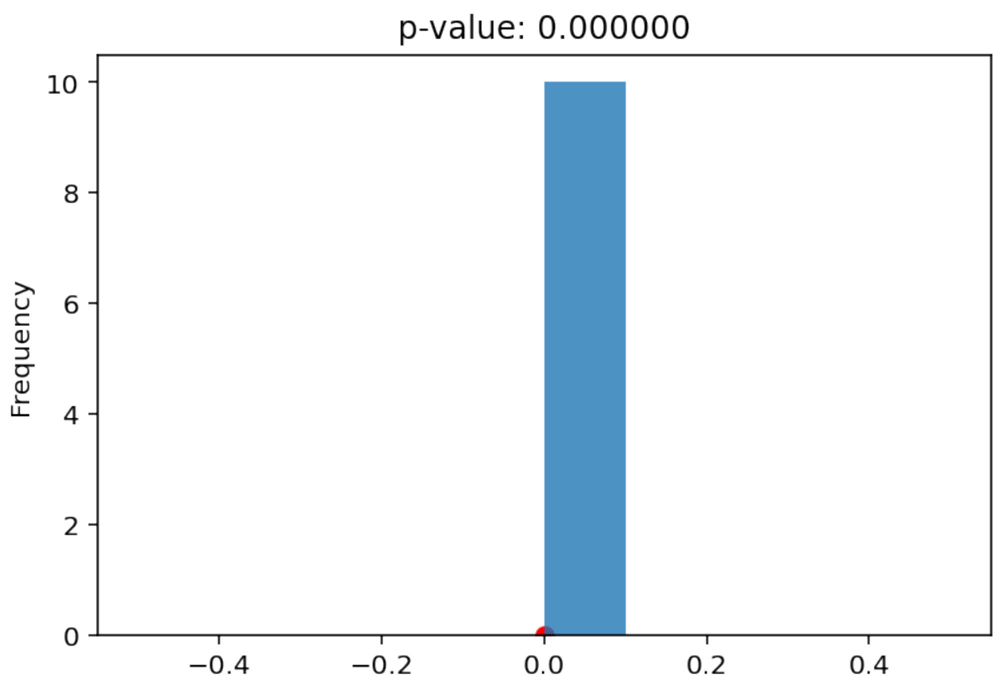
Conducting a permutation test to assess whether the missingness of the gender column is dependent on the officers gender:

[21]:
```
mcar2_nypd = nypd[['complainant_gender', 'board_disposition', 'allegation',␣
 ↪'mos_gender']]
distr = (
    mcar2_nypd
    .assign(is_null=mcar2_nypd['complainant_gender'].isnull())
    .pivot_table(index='is_null', columns='mos_gender', aggfunc='size')
)
distr = (distr.T / distr.sum(axis=1)).T
distr


n_repetitions = 100

tvds = []
for _ in range(n_repetitions):

    # shuffle the board_disposition column
    shuffled_col = (
        mcar2_nypd['allegation']
```

```
        .sample(replace=False, frac=1)
        .reset_index(drop=True)
    )

    # put them in a table
    shuffled = (
        mcar2_nypd
        .assign(**{
            'mos_gender': shuffled_col,
            'is_null': mcar2_nypd['complainant_gender'].isnull()
        })
    )

    # compute the tvd
    shuffled = (
        shuffled
        .pivot_table(index='is_null', columns='mos_gender', aggfunc='size')
        .apply(lambda x:x / x.sum(), axis=1)
    )

    tvd = shuffled.diff().iloc[-1].abs().sum() / 2
    # add it to the list of results

    tvds.append(tvd)


obs = distr.diff().iloc[-1].abs().sum() / 2


# Calculating p-value
pval = np.mean(tvds > obs)
pval
```

[21]: 0.0

[22]:
```
# Plotting graph to visualize p-value

pd.Series(tvds).plot(kind='hist', density=True, alpha=0.8, title='p-value: %f'␣
 ↪% pval)
plt.scatter(obs, 0, color='red', s=40);
```



Conclusion Since the p value we obtained is again 0.83, we can conclude that the missingess of complainant_gender does depend on officer's gender. Some civilians who made a complaint may have chosen to left the gender column blank to avoid being identified.

3.0.3 Hypothesis Test

3.0.4 Question 1: Permutation test

Are the complaints of women more successful than men (for the same allegations)?

Conducting a permutation test to check if females have more succesful complaint outcomes than men for each allegation We decided on the test statistic which sums the difference in proportions because this made sense to us and we were able to group by each allegation and then compare the difference in proportions.

Calculating observed value for question 1:

[23]:
```
# Calculating total diff in proportion of succesful complaints for each␣
↪allegation for males and females and summing the differences

observed_summed_proportion = 0
for allegation in list(only_m_and_f['allegation'].unique()):

    df = only_m_and_f[only_m_and_f['allegation'] == allegation]
    females = df[df['complainant_gender'] == 'Female']
```

```
    males = df[df['complainant_gender'] == 'Male']

    # making sure there is at least 1 instance of this allegation in male and␣
 ↪female
    if (females.shape[0] > 0 ) and (males.shape[0] > 0 ):
        females_succesful_proportion = females[females['board_disposition'] ==␣
 ↪'Substantiated'].shape[0]/females.shape[0]
        male_succesful_proportion = males[males['board_disposition'] ==␣
 ↪'Substantiated'].shape[0]/males.shape[0]
        # Calculating difference in female success rate - male success rate
        observed_summed_proportion += (females_succesful_proportion -␣
 ↪male_succesful_proportion)
    else:
        observed_summed_proportion += 0

observed_summed_proportion
```

[23]: 0.36695513209341596

Shuffling and conducting permutation test for question 1:

[24]:

```
# Shuffling male and female labels and calculating total diff in proportions by␣
 ↪conducting a permutation test



N = 1000
means = []
shuffled = only_m_and_f.copy()[['officer_id','complaint_id',␣
 ↪'complainant_gender', 'allegation', 'board_disposition']]
shuffled_genders = shuffled['complainant_gender'].values

for _ in range(N):

    # shuffle the genders
    shuffled_genders = np.random.permutation(shuffled_genders)
    shuffled['complainant_gender'] = shuffled_genders

    summed_means = 0
    for allegation in list(shuffled['allegation'].unique()):

        df = shuffled[shuffled['allegation'] == allegation]
        females = df[df['complainant_gender'] == 'Female']
        males = df[df['complainant_gender'] == 'Male']

        # making sure there is at least 1 instance of this allegation in male␣
 ↪and female
        if (females.shape[0] > 0 ) and (males.shape[0] > 0 ):
```

```
            females_succesful_proportion = females[females['board_disposition']␣
↪== 'Substantiated'].shape[0]/females.shape[0]
            male_succesful_proportion = males[males['board_disposition'] ==␣
↪'Substantiated'].shape[0]/males.shape[0]
            # Calculating difference in female success rate - male success rate
            summed_means += females_succesful_proportion -␣
↪male_succesful_proportion
        else:
            summed_means += 0

    means.append(summed_means)

means = pd.Series(means)
```

[25]:
```
# Calculating p-value for summed proportion differences compared to observed

pval_means = (means >= observed_summed_proportion).sum() / N
pval_means
```

[25]: 0.282

Conclusion: We fail to reject the null hypothesis at the significance level of 0.05 because the p-value is greater than 0.05. Therefore, we cannot conclude that there women are more likely to have a succesful complaint than males. This might suggest that the board disposition is usually fair and not biased towards women.

3.0.5 Question 2: Permutation test

Are white-officers more likely to be exonerated or unsubstantiated?

Conducting a permutation test to check if white-officers are more likely to get off the hook (exonerated or unsubstantiated) We chose the test statistic which takes the difference in means into consideration because this would clearly highlight whether there is any real difference or not. We also chose this since we are summing the number of white officers and non white officers so our data is quantitative.

[26]:
```
#Calculating Total Diff
nypd_white = nypd[nypd['mos_ethnicity'] == 'White']
nypd_non_white = nypd[nypd['mos_ethnicity'] != 'White']
```

[27]:
```
# Calculating observed difference in proportions for non white officers who get␣
↪off the hook
# and white officers who get off the hook
test = nypd_white.copy()
cnt = test.pivot_table(index='mos_ethnicity', columns='board_disposition',␣
↪aggfunc='size')
distr = cnt/test.shape[0] #In proportion
```

```
white = 1 - distr['Substantiated'].iloc[0]

tests = nypd_non_white.copy()
cnts = tests.pivot_table(index='mos_ethnicity', columns='board_disposition',␣
↪aggfunc='size')
distrs = cnts/tests.shape[0] #In proportion
val = 0
for i in distrs['Substantiated']:
    val += i
non_white = 1 - val

observed_mean = white - non_white
observed_mean
```

[27]: 0.013362055475429568

Shuffling and conducting permutation test for question 2:

[28]:
```
N = 100
mean = []
shuffle = nypd.copy()[['mos_ethnicity','board_disposition']]
shuffle_ethnicity = shuffle['mos_ethnicity'].values

for _ in range(N):
    #Shuffle the ethnicity
    shuffle_ethnicity = np.random.permutation(shuffle_ethnicity)
    shuffle['mos_ethnicity'] = shuffle_ethnicity

    nypd_white = shuffle[shuffle['mos_ethnicity'] == 'White']
    nypd_non_white = shuffle[shuffle['mos_ethnicity'] != 'White'] #this␣
↪includes nan filled with 0 and unknown

    nypd_white_mean = 1 - (nypd_white[nypd_white['board_disposition'] ==␣
↪'Substantiated'].shape[0] / nypd_white.shape[0])
    nypd_non_white_mean = 1 -␣
↪(nypd_non_white[nypd_non_white['board_disposition'] == 'Substantiated'].
↪shape[0] / nypd_non_white.shape[0])

    diff = (nypd_white_mean - nypd_non_white_mean)

    mean.append(diff)

mean = pd.Series(mean)
```

[29]:
```
# P value
pval_mean = (mean >= observed_mean).sum() / N
pval_mean
```

```
[29]: 0.0
```

Conclusion: We strongly reject the null hypothesis because our p-value is lower than 0.05. We can conclude that there is a difference in the number of white-officers who get exonerated or unsubstantiated compared to the non white officers. This goes against are conclusion that the board is probably unbiased from the previous permutation test and suggests that there could be some amount of bias. On the other hand, it is still possible that there are other confounding factors which led us to this result. Moreover, there is also the strong possibility that the dispositions were correct and by coincidence, the white-officers did not do anything wrong and were wrongly accused.