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](#).

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

About the dataset

This data set comes from a database of 12,056 civilian complaints filed against 3996 New York City police officers (all of the allegations that did not occur concluded by the investigators are excluded in this dataset, while there are still some unsubstantiated records that cannot be confirmed to happen or not). I will use the `CCRB Data Layout Table.xlsx` as a reference to analyze this dataset, which shows me how can I simplify my dataset.
In this project, I will analyze the missingness of every columns and try to find whether their missingness are dependent on other columns if I think they are **not** NMAR.
Also, I will try to discover whether the disposition of the officers will be affected by the outcome of the contact between office and complainant (Arrest/Summon or not).

Cleaning and EDA

- **Data Cleaning**
  - **Check every columns and try to find whether there are some data thath should be missing with `NaN`. e.g. 'Not Described' Gender should be labeled as missing here.**
  - **Dataset Simplification: According to the `CCRB Data Layout Table.xlsx`, I find that there are lots of detailed information that I do not need, so I simplify or remove them. Related columns: `rank_abbrev_now`, `rank_abbrev_incident`, `outcome_description` and `board_disposition`.**
  - **New Columns: I create 3 new columns to get information that will be used in the EDA and hypothesis statistical test from the existing columns: `same_ethnicity`, `isna_age`, `isna_reason`**
- **EDA**
  - **Univariate Analysis: I analyze the column `year_received` to check the trend of the frequency of the complaints. I also check the column `allegation` in order to find the categories that are complained most frequently.**
  - **Bivariate Analysis: Here I focus on two columns `outcome_description` and column `board_disposition` to check whether the distribution of the disposition of officers are similar given different outcome of the complainants. And my hypothesis test is based on this bivariate analysis.**
  - **Aggregation Analysis: I will check the relationship between the most 10 frequently unwanted actions (from the univariate analysis) above and the year when those complains are received.**

Assessment of Missingness

In this part I will firstly check whether one column is NMAR and give my reasons. Then I analyze two columns (`complainant_age_incident` and `contact_reason`) use permutation test to check whether there missingness depend on other columns.

Hypothesis Test

In this part I will do a further study on my question posted on the bivariate analysis --- whether the distribution of the disposition of officers are similar given different outcome of the complainants. Here I conduct hypothesis test to prove this and find that the distribution of the disposition of officers do be affected by the outcome of the complainants themselves.

Code

In [2]:

```
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
# Show all dataframe columns
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 20)
```

In [4]:

```
# Read the csv to dataframe
DF = pd.read_csv('allegations_202007271729.csv')
```

Dataset Basic Information

Here I will give some basic code showing the information of this dataset.

- **1. Number of all records, Number of unique officer id, Number of complaint id.**
- **1. Different kinds of dispositions.**
- **1. Missing Columns**

Now we find that the one complaint id, and the unique id are not equal to the number of records. We can see that one complaints can contain many records if the user complained about many issues (there are lots of categories of complaint) with one officer.

In [5]:

```
print('Number of records:', DF.shape[0])
print('Number of complaint id:', DF['complaint_id'].nunique())
print('Number of unique officer id:', DF['unique_mos_id'].nunique())
```

```
Number of records: 33358
Number of complaint id: 12056
Number of unique officer id: 3996
```

This 'board_disposition' column has lots of values, we can contract them into 3 categories (`Substantiated`, `Exonerated`, `Unsubstantiated`) in the cleaning section according to the CCRB Table xlsx file which only shows 3 dispositions. Also, the detailed information on the disposition is not what we need to analyze here.

In [32]:

```
DF['board_disposition'].value_counts()
```

Out[32]:

```
Unsubstantiated                                15448
Exonerated                                      9609
Substantiated (Charges)                         3796
Substantiated (Formalized Training)             1033
Substantiated (Command Discipline A)             964
Substantiated (Command Discipline)               851
Substantiated (Command Discipline B)             789
Substantiated (Command Lvl Instructions)         454
Substantiated (Instructions)                     248
Substantiated (No Recommendations)               165
Substantiated (MOS Unidentified)                   1
Name: board_disposition, dtype: int64
```

Now we find that the missing columns and the proportion of the missiness of this dataset. This series will be used in the `Assessment of Missingness` Part (Some numbers may be changed after the data cleaning)

In [6]:

```
# Find the missining columns, and we can find that there are a lots of missingness on the three columns:
DF.isna().sum().sort_values(ascending=False).head(10)
```

Out[6]:

```
complainant_age_incident    4812
complainant_ethnicity       4464
complainant_gender          4195
command_at_incident         1544
contact_reason               199
outcome_description           56
precinct                      24
allegation                     1
unique_mos_id                  0
rank_incident                  0
dtype: int64
```

Cleaning and EDA

I will replace some data that are actually missing (with unresonable values) by np.NaN. Also, I will try to adjust some data in certain columns to simplify my analysis

Part 1 Clean the Data:

- **Check whether there are missing data that isn't labeled as missing (Numerical).**

- **For the numerical data (use `DF.dtypes` to check the data types), we can use value_counts to check whether there are strange number (I will use `mos_age_incident`, `month_received` as examples here). I use `pd.value_counts()` here to observe those numerical columns.**
- **After checking all of the numerical columns in this method, I find that `complaint_age_incident` has some negative values (-1, -4301) and 2 records with age 101, I will set them to np.NaN.**
- **Also, column `precinctt` also contains some strange values 0 and 1000. This percinctt is a nominal data. New York do not have 1000th pertinent and 0th pertinent. i.e. The pertinent should in range(1, 123). Thus, I will replace them with np.NaN.**

In [7]:

```
DF['complainant_gender'].value_counts()
```

Out[7]:

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

In [13]:

```
'''
mos_age_incident column
all of the indices is in range of [20,  60], and there is no strange data,
which is reasonable.
'''
DF['mos_age_incident'].value_counts().head()
```

Out[13]:

```
30    2336
28    2240
29    2229
31    2199
27    2100
Name: mos_age_incident, dtype: int64
```

In [12]:

```
'''
month_received column
all of the indeices is in range of [1, 12], and there is no strange data,
which is reasonable.
'''
DF['month_received'].value_counts().head()
```

Out[12]:

```
3    3154
8    2980
5    2968
9    2966
4    2881
Name: month_received, dtype: int64
```

In [14]:

```
'''
complaint_age_incident column
Substitute the data with negative value or over 100 to np.NaN
'''
DF.loc[((DF['complainant_age_incident'] > 100 )|
       (DF['complainant_age_incident'] < 0)), 'complainant_age_incident'] = np.NaN
# Check again with the value_counts method metioned above
DF['complainant_age_incident'].value_counts().sort_values().head()
```

Out[14]:

```
84.0    1
1.0     1
88.0    1
83.0    1
90.0    1
Name: complainant_age_incident, dtype: int64
```

In [15]:

```
'''
complaint_age_incident column
Substitute the data with 0 or 1000 (not in range(1, 123) )to np.NaN
'''

DF.loc[((DF['precinct'] < 1) | (DF['precinct'] > 123)), 'precinct'] = np.NaN
DF['precinct'].value_counts().sort_values().head()
```

Out[15]:

```
22.0      12
111.0     37
17.0      76
66.0      89
123.0    112
Name: precinct, dtype: int64
```

- **Check whether there are missing data that isn't labeled as missing (Categorical). As to the categorical data, I will only focus on the following columns that may be used in my analysis: `rank_abbrev_incident`, `rank_abbrev_now`, `mos_ethnicity`, `mos_gender`, `complainant_ethnicity`, `complainant_gender`, `fado_type`, `allegation`, `contact_reason`, `outcome_description`, `board_disposition` I can still use the `value_counts()` to check whether there are unresonable categories that should be replaced by np.NaN**
  - **Then I find that `complainant_gender` has some records with value `'Not described'`, I will replace them with np.NaN**
  - **The `complainant_ethnicity` has records with `Unkown` and `Refused`, I will replace them with np.NaN**

In [16]:

```
'''
complaint_gender column
Substitute the data with Not described to np.NaN
'''
DF.loc[DF['complainant_gender'] == 'Not described', 'complainant_gender'] = np.NaN
DF['complainant_gender'].value_counts()
```

Out[16]:

```
Male                     24058
Female                    5021
Transwoman (MTF)            20
Transman (FTM)               5
Gender non-conforming        2
Name: complainant_gender, dtype: int64
```

In [17]:

```
'''
complaint_ethnicity column
Substitute the data with Refused or Unknown to np.NaN
'''
DF.loc[((DF['complainant_ethnicity'] == 'Unknown') | (DF['complainant_ethnicity'] == 'Re
fused')),
                                                  'complainant_ethnicity'] = np.NaN
DF['complainant_ethnicity'].value_counts()
```

Out[17]:

```
Black              17114
Hispanic            6424
White               2783
Other Race           677
Asian                532
American Indian       64
Name: complainant_ethnicity, dtype: int64
```

- **After finding the 'missing data' that isn't labeled as missing, I will also concat some categories in a certain column to one category. Here are 4 columns need to be simplified: `rank_abbrev_now`, `rank_abbrev_incident`, `outcome_description` and `board_disposition`. The reason behind this simplification comes from the `CCRB Data Layout Table.xlsx` saying the following information:**
  - **board_disposition has only three results: Substantiated, Exonerated, Unsubstantiated. In the original dataset, this column contains some details about how the officer's behavior is substantiated, which are not needed, so I will replace them with only 'Substantiated'**
  - **Police officer have 3 kinds of abbreviations (POM, POF, PO), I will simplified them into one abbreviation PO, also I will drop the columns 'rank_now', 'rank_incident' columns since they contain the same information as the abbreviation**
  - **Also, I am not care about the details of the arrest/summon. I will combine all the arreste/summon outcome into only one category: `Arrested or Summoned`.**

In [18]:

```
'''
board_disposition column
Repalce the records with details like Substantiated (Command Discipline A) by only Substa
ntiated to
simplify the analysis
'''

DF['board_disposition']= DF['board_disposition'].str.replace(r'Substantiated .+', 'Subst
antiated',
                                                              regex=True)
DF['board_disposition'].value_counts()
```

Out[18]:

```
Unsubstantiated    15448
Exonerated          9609
Substantiated       8301
Name: board_disposition, dtype: int64
```

In [19]:

```
'''
rank_abbrev_now, rank_abbrev_incident columns
Repalce the POM, POF by PO, since all of these three are the abbreviations of the Police
officer rank.
'''
DF['rank_abbrev_now']= DF['rank_abbrev_now'].str.replace(r'POM|POF', 'PO',
                                                              regex=True)

DF['rank_abbrev_incident']= DF['rank_abbrev_incident'].str.replace(r'POM|POF', 'PO',
                                                              regex=True)
# Drop the unused columns
DF.drop(['rank_incident', 'rank_now'], axis=1, inplace=True)
DF['rank_abbrev_incident'].value_counts()
```

Out[19]:

```
PO      22509
SGT      5701
DT3      2712
LT       1264
DTS       330
DT2       195
CPT       182
SDS       128
```

```
SSA       105
DI         96
DET        50
INS        27
LSA        24
DT1        20
LCD        13
DC          2
Name: rank_abbrev_incident, dtype: int64
```

In [124]:

```
'''
outcome_description column
Simplify the arrest/summon records with details by Arrested or Summoned.
'''
DF.loc[(~(DF['outcome_description'] == 'No arrest made or summons issued')),
       'outcome_description'] = 'Arrested or Summoned'
DF['outcome_description'].value_counts()
```

Out[124]:

```
Arrested or Summoned                20536
No arrest made or summons issued    12821
Name: outcome_description, dtype: int64
```

- **Finally, I will create new columns to get some more information that will be used in the EDA and hypothesis statistical test from the existing columns.**

  New column `same_ethnicity`: This column contains only `True` or `False`, if the officer's ethnicity and the complainant's ethnicity are the same, this column is `True`, and vice versa. Also, if the complainant's ethnicity is missing, I will set the value in `same_gender` column to be `False`.

  New column `isna_age`: This column contains only boolean `True` or `False`, if the complainant's age is missing, I will set the value to be `True`, and vise versa.

  New column `isna_reason`: This column contains only boolean `True` or `False`, if the contact_reason column is missing, I will set the value to be `True`, and vise versa.

In [37]:

```
DF['same_ethnicity'] = (DF['complainant_ethnicity'] == DF['mos_ethnicity'])
DF['same_ethnicity']
'''
isna_age column
Whether the column complainant_age_incident record is np.NaN
'''
DF['isna_age'] = DF['complainant_age_incident'].isna()
'''
isna_reason column
'''
DF['isna_reason'] = DF['contact_reason'].isna()
```

Data cleaning is done, I will do the EDA in the following cells

Univariate Analysis

In this part I will show you how I analysis univariate data:

- **year_received: This column contains the Year the complaint was received by CCRB from Sept. 1985 to Jan. 2020. Before analyzing this data, I will remove the data of 1985 and 2020 because the data of these two years is not integrated. The analysis of this data will show you the trend of the frequency of the complaints.**

  Also, the dataset contains lots of same complaint_id reporting multiple issues, I will remove this factor by 'select distinct' compliant_id before I analyze this year_received data.

Finally I find that the average number of compliants per year is about 355 cases. And such kinds of compliants increased a lot from 2000 to 2005. However, **we cannot judge that people are more dissatisfied with the police, beacuse we do not know the total number of cases here.**

It is possible that the number of complians increase only, because there are more 'contacts' between the police and the civilians, which increase the cardinal number of compliants.

- **allegation:** This column contains the specific category of complains. I analyze this column in order to find the allegation categories that are complained most frequently. i.e. The 'disgusted' actions the police take most frequently.

  The result of this analysis shows that `Physical Force` and discourteous `word` are the most unwanted actions taken by the police, which take a huge proportions of the allegations. (You can check my bar plot in the following cell)

year_received column

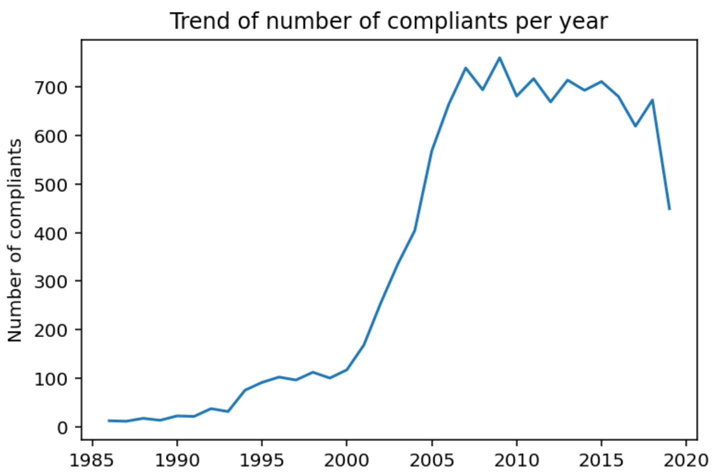
In [216]:

```
'''
year_received column Univariate Analysis
Before the analysis:
"Select distinct" complaints id, and ignore the data of 1985 and 2020.
'''
# Remove the duplicated complaint id for this univariate analysis
needed = DF.drop_duplicates(subset='complaint_id')
# Remvoe the data of 1985 and 2020
needed = needed.loc[(~(needed['year_received'] == 1985) & ~(needed['year_received'] == 2
020)),
                    'year_received']
# Use value_counts() to get the number of compliants for each year
num_compliants = needed.value_counts().sort_index()
print('Average number of compliants per year: ', num_compliants.mean())
num_compliants.plot(kind='line', title='Trend of number of compliants per year',
                    ylabel='Number of compliants')
```

```
Average number of compliants per year:  354.4117647058824
```

Out[216]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e49f3b8>
```



allegation column

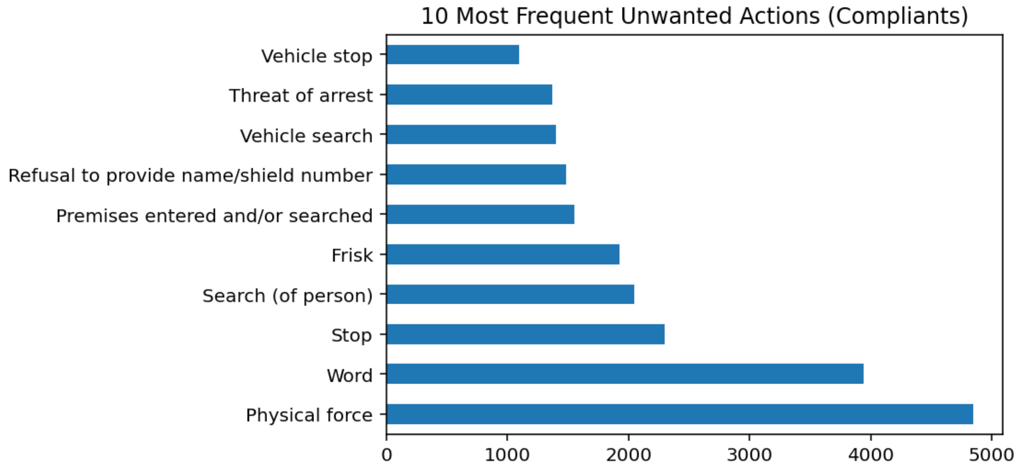
In [229]:

```
'''
```

```
allegation column Univariate Analysis
count the frequency for every kinds of allegations.
'''
needed = DF['allegation']
most_freq = needed.value_counts().sort_values(ascending=False).iloc[:10]
most_freq.plot(kind='barh', title='10 Most Frequent Unwanted Actions (Compliants)')
```

Out[229]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e4d7d18>
```



Bivariate Analysis

In this part, I will focus on a pair of columns to analyze. Let's focus on whether the outcome of the contact between office and complainant --- column `outcome_description` has any relationship with the disposition of the officcers --- column `board_disposition` . The reason why I pose this question is beacuse I am wondering whether the compliants from the civilians breaking the laws will be "belittled" (more unsubstantiated dispositions).

After this bivariate test, I have got the bar plot and the TVD as the statistic. From the bar plot and the TVD, we can see that there are some difference between the distributions of the disposition result whether the complainants are punished themselves. We can see that the officer will be more likely to be exonerated if the complainants themselves are arrested.
We can conduct the hypothesis test later to confirm this conjecture in the final part of this project.

In [125]:

```
'''
outcome_description, board_disposition Bivariate Analysis
'''
# Generally, we can condense the outcome into two categories [No arrest and summon, Arres
ted or Summoned]
needed = (DF['outcome_description'] == 'No arrest made or summons issued')
needed = needed.replace({True:'No arrest made or summons issued', False:'Arrested or Summ
oned'})
# Copy the disposition columns twice to get count the frequency later.
cc = pd.concat([needed, DF['board_disposition']], axis=1)
cc.columns = ['outcome_description', 'board_disposition']
table = cc.groupby(['outcome_description'])['board_disposition'].value_counts(normalize=
True)
# Normalization
result = pd.DataFrame(table)
result.columns = ['disposition_counts']
table = result.pivot_table(index='board_disposition',
                    columns='outcome_description',
                    values='disposition_counts')
table
```

Out[125]:

outcome_description	Arrested or Summoned	No arrest made or summons issued

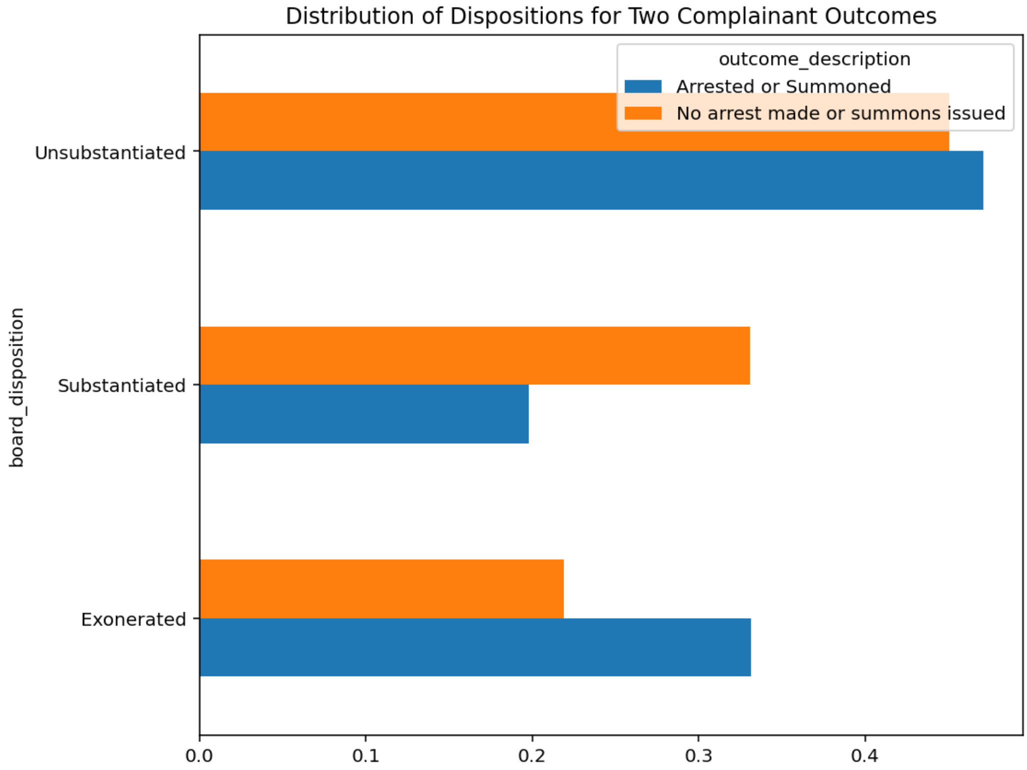
outcome_description board_disposition	Arrested or Summoned	No arrest made or summons issued
Exonerated	0.331321	0.218782
Substantiated	0.197750	0.330707
Unsubstantiated	0.470929	0.450511

In [126]:

```
# Get the bar plot
table.plot(kind='barh', figsize=(8,7),title='Distribution of Dispositions for Two Complai
nant Outcomes')
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a8a34c0>
```



In [127]:

```
# CaLculate TVD
TVD = table.diff(axis=1).abs().sum(axis=0).iloc[-1] / 2
TVD
```

Out[127]:

```
0.13295714094769345
```

Aggregation Analysis

In this part, I will focus on the aggregation statistics to identify possible associations.

- **year_received, month_received and allegation : I will check the relationship between the most 10 frequently unwanted actions (from the univariate analysis) above and the year those complaints received. Since both of these two columns are categorical data, I will use `pivot table` and the `bar plot` to show the result here.**

  As you can see from the plots below: We can see that the proportion of `Physical force` is reduced from 1999 to 2019, while there are more kinds of 'unwanted actions' occur (like `Search (of person)`, `Frisk`). Also, using discourteous `word` maintains a high proportion of those unwatned actions.

In [332]:

```
'''
year_received allegation Aggregation Analysis
Firstly, I need to filter the allegations not in the 10 most frequent unwanted actions fr
om the dataframe
(using the result above).
Then I will plot the pivot table (use an extra column "month_received" to count)
to count the number of cases and conduct the normalization.
'''
needed = DF.loc[DF['allegation'].isin(most_freq.index.values), ['year_received', 'month_
received',
                                                              'allegation']]
table = needed.pivot_table(columns='allegation', index='year_received', values='month_re
ceived',
                  aggfunc='count').fillna(0)
# drop 1997, 1998 and 2020, since the dataset is too small.
table = table.drop([1997, 1998, 2020])
# Normalization
result = table.div(table.sum(axis=1), axis=0)
result.head()
```

Out[332]:

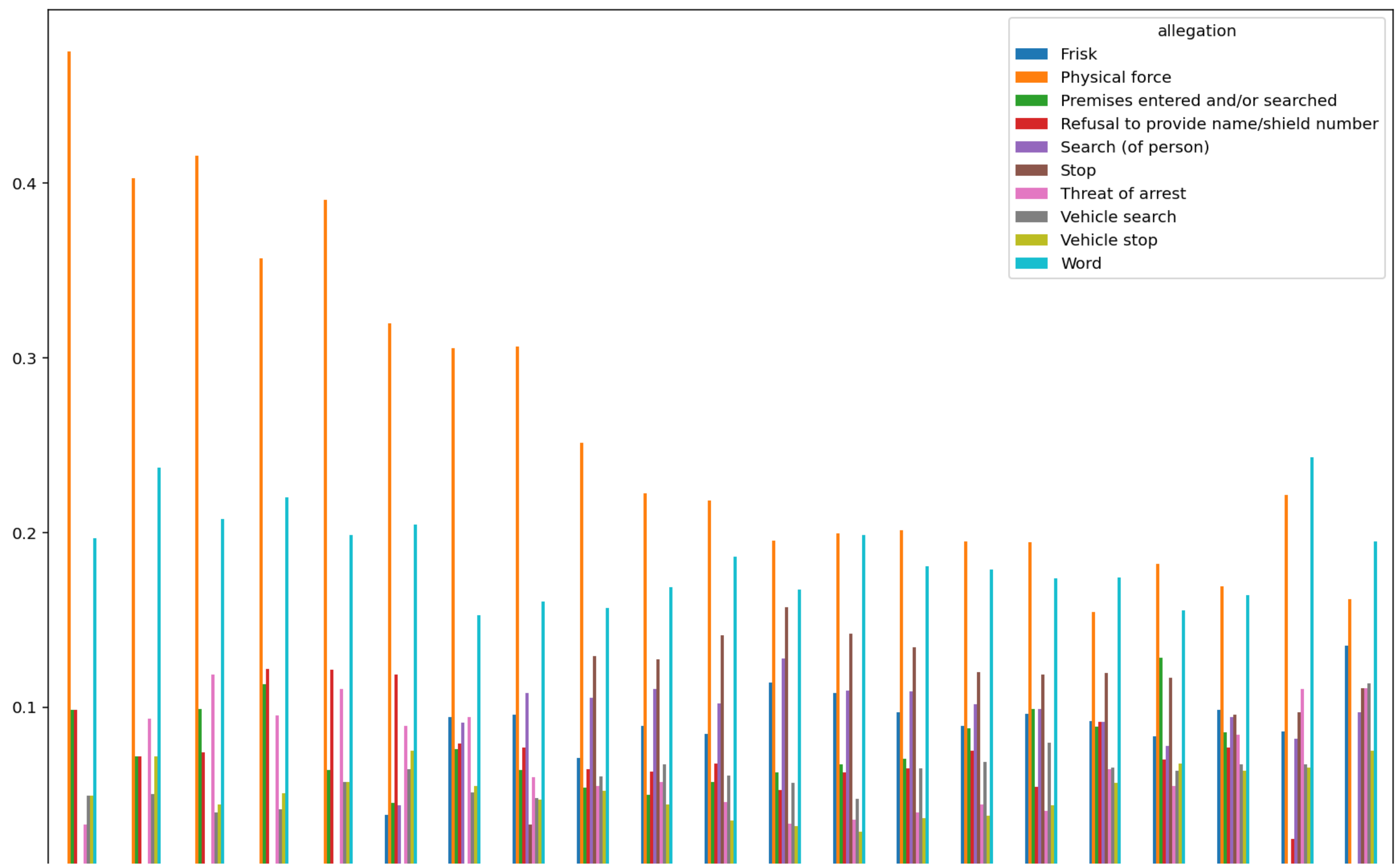
allegation	Frisk	Physical force	Premises entered and/or searched	Refusal to provide name/shield number	Search (of person)	Stop	Threat of arrest	Vehicle search	Vehicle stop	Word
year_received										
1999	0.0	0.475410	0.098361	0.098361	0.0	0.0	0.032787	0.049180	0.049180	0.196721
2000	0.0	0.402878	0.071942	0.071942	0.0	0.0	0.093525	0.050360	0.071942	0.237410
2001	0.0	0.415842	0.099010	0.074257	0.0	0.0	0.118812	0.039604	0.044554	0.207921
2002	0.0	0.357143	0.113095	0.122024	0.0	0.0	0.095238	0.041667	0.050595	0.220238
2003	0.0	0.390728	0.064018	0.121413	0.0	0.0	0.110375	0.057395	0.057395	0.198675

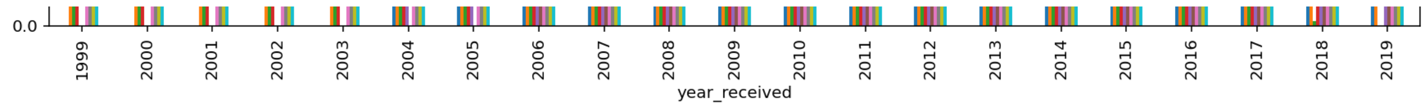
In [333]:

```
result.plot(kind='bar', figsize=(15,10))
```

Out[333]:

<matplotlib.axes._subplots.AxesSubplot at 0x20fe4c28>





Interesting Aggregates

In this part I choose `allegation`, `fado_type`, `board_disposition` to conduct the aggregation analysis. I will try to find what kinds of complaint will turn out to be considered as the violation of rules.

In [273]:

```
DF.groupby(['fado_type', 'allegation'])['board_disposition'].count()
```

Out[273]:

```
fado_type            allegation
Abuse of Authority   Arrest/D. A. T.                           6
                     Arrest/Onlooker                           4
                     Body Cavity Searches                      3
                     Detention                                19
                     Electronic device information deletion   11
                                                             ...
Offensive Language   Physical disability                      15
                     Race                                    307
                     Religion                                 14
                     Sexual orientation                       78
                     White                                     8
Name: board_disposition, Length: 118, dtype: int64
```

In [276]:

```
DF.isna().sum().sort_values(ascending=False).iloc[:10]
```

Out[276]:

```
complainant_age_incident    4820
complainant_ethnicity       4464
complainant_gender          4252
command_at_incident         1544
contact_reason               199
outcome_description           56
precinct                      48
allegation                     1
unique_mos_id                  0
mos_gender                     0
dtype: int64
```

Assessment of Missingness

Here is the layout of this part of content:

- **1. Check the missing columns**
- **1. Missness Analysis**

Here are the missing columns, I will analyze them one by one in the cells below

In [320]:

```
DF.isna().sum().sort_values(ascending=False).iloc[:10]
```

Out[320]:

```
complainant_ethnicity       5763
complainant_age_incident    4819
complainant_gender          4251
command_at_incident         1543
contact_reason               199
outcome_description           56
precinct                      48
unique_mos_id                  0
```

```
unique_mos_id                 0
mos_gender                    0
board_disposition             0
dtype: int64
```

complainant_age_incident: It is possible to consider this kind of missing to be NMAR.Most of the complainants' age are about 20-40. One may choose not to report there ages on there own will. *I will also conduct the permutation tests to check whether the missingness of this column depends on other columns.*
However, we cannot exclude NMAR from all the possibilities. There may be some "accidents" during the analysis.

Check the dependency on column `year_received`. I am worried about whether the missiness is because of the lag in technology in the early years.

Null Hypothesis: The distribution of year is similar for null/not-null.
Alternative Hypothesis: The distribution of year is similar for null/not-null. (TVD is small)
Since the two columns are both categorical data. We use TVD as the test-statistic.

Check the dependency on column `board_disposition`. I am worried about whether the missiness is because of the lag in technology in the early years.

Null Hypothesis: The distribution of year is similar for null/not-null.
Alternative Hypothesis: The distribution of year is similar for null/not-null. (TVD is small)
Since the two columns are both categorical data. We use TVD as the test-statistic.

Conclusion: The p-value of these two hypothesis tests are both 0.0. We can reject the null hypothesis, and conclude that the `complainant_age_incident` is most likely to be MAR depending on `year_received` and `board_disposition` column, if it is not NMAR.
However, I have checked most of the columns in the dataset and find the p-value all close to 0, I have not met this situation before, maybe that is because the missingness of `complainant_age_incident` column is truly NMAR so I cannot get any MCAR conclusion here.

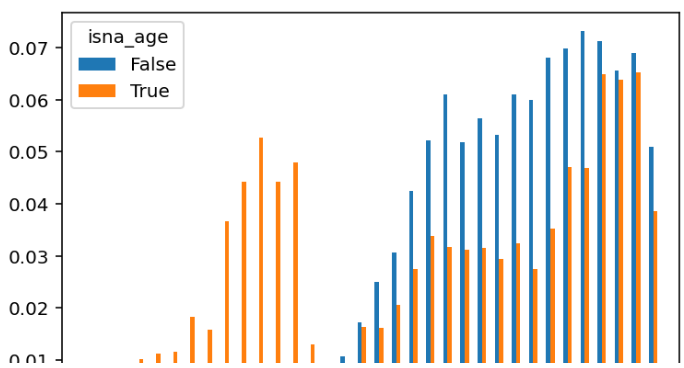
Hypothesis Test on Column `year_recieved`

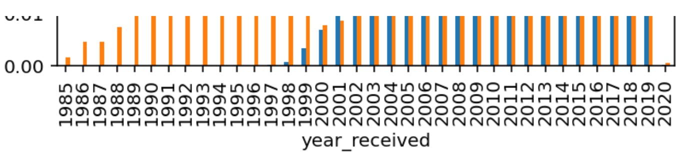
In [49]:

```
'''
Get the observed statistic from the original DF
Firstly, let's get the bar plot firstly
'''
obs = pd.DataFrame(DF.groupby(['isna_age'])['year_received'].value_counts(normalize=True
))
obs.columns=['year_proportion']
table = obs.pivot_table(index='year_received', columns='isna_age', values='year_proporti
on').fillna(0)
table.plot(kind='bar')
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19dc2718>
```





In [58]:

```
'''
Then we calculate the tvd use the pivot table above.
'''
obs_TVD = table.diff(axis=1).abs().iloc[:, -1].sum() / 2
obs_TVD
```

Out[58]:

```
0.3209150702172163
```

In [73]:

```
'''
Permutation Test
'''
TVDs = []
for _ in range(1000):
    DF['sampled'] = np.random.permutation(DF['isna_age'])
    sample =  DF.pivot_table(index='year_received',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
    TVDs.append(sample.diff(axis=1).iloc[:, -1].abs().sum() / 2)

TVDs = pd.Series(TVDs)
TVDs.head()
```

Out[73]:

```
0    0.030511
1    0.032313
2    0.028140
3    0.023930
4    0.032225
dtype: float64
```

In [75]:

```
'''
Get the p-value, and give the conclusion
'''
p_value = (TVDs >= obs_TVD).mean()
p_value
```

Out[75]:

```
0.0
```

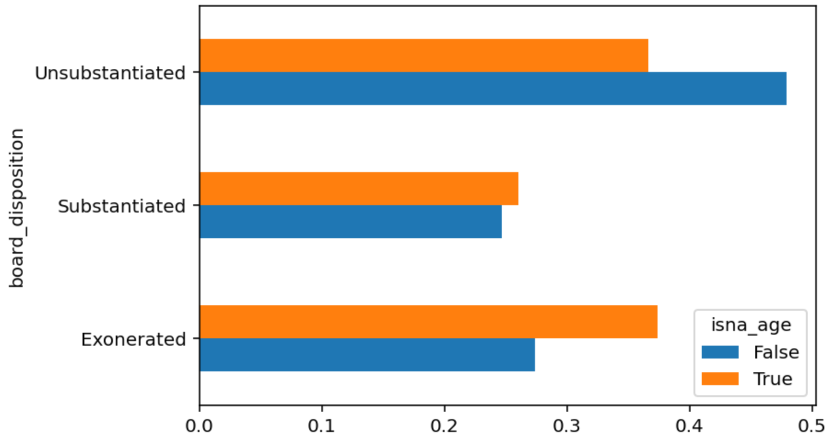
Hypothesis Test on Column `board_disposition`

In [114]:

```
'''
Get the observed statistic from the original DF
Firstly, let's get the bar plot firstly
'''
obs = pd.DataFrame(DF.groupby(['isna_age'])['board_disposition'].value_counts(normalize=
True))
obs.columns=['disposition_proportion']
table = obs.pivot_table(index='board_disposition',
                        columns='isna_age',
                        values='disposition_proportion').fillna(0)
table.plot(kind='barh')
```

Out[114]:

<matplotlib.axes._subplots.AxesSubplot at 0x1a857bc8>



In [115]:

```
'''
Then we calculate the tvd use the pivot table above.
'''
obs_TVD = table.diff(axis=1).iloc[:, -1].abs().sum() / 2
obs_TVD
```

Out[115]:

0.11328340257686559

In [116]:

```
'''
Permutation Test
'''
TVDs = []
for _ in range(1000):
    DF['sampled'] = np.random.permutation(DF['isna_age'])
    sample =  DF.pivot_table(index='board_disposition',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
    TVDs.append(sample.diff(axis=1).iloc[:, -1].abs().sum() / 2)

(TVDs >= obs_TVD).mean()
```

Out[116]:

0.0

- **complainant_gender: It is reasonable to consider this kind of missingness to be NMAR. There are some genders (like transgender, or non-conforming gender) that a civilian may choose not to report. Such kinds of situations will lead to missingness.**

In [321]:

```
DF['complainant_gender'].value_counts()
```

Out[321]:

```
Male                        24058
Female                       5021
Transwoman (MTF)               20
Transman (FTM)                  5
Gender non-conforming           2
Name: complainant_gender, dtype: int64
```

- **complainant_ethnicity: This kind of missing is NMAR just like what I did in data cleaning step. There are someone who do not sure about there ethnicities, also there are some civilians refuse to report there ethnicities. Such kinds of situations will lead to missingness.**

- **conctact_reason: I think this kind of missing is MCAR, I will conduct a permutation test to check some columns here.**

  Check the dependency on column `allegation`: There may be some kinds of allegations having specified contact_reasons. I guess this column may affect the missingness of the `allegation` column. Null Hypothesis: The distribution of year is similar for null/not-null.
  Alternative Hypothesis: The distribution of year is similar for null/not-null. (TVD is small) Since the two columns are both categorical data. We use TVD as the test-statistic.

  Check the dependency on column `rank_abbrev_incident`: There should not have any difference between these 2 columns according to the common sense. I want to check whether the strange situations in the missingness analysis on column `complainant_age_incident` happens again (all p-value equals to 0.0).
  Null Hypothesis: The distribution of year is similar for null/not-null.
  Alternative Hypothesis: The distribution of year is similar for null/not-null. (TVD is small) Since the two columns are both categorical data. We use TVD as the test-statistic.

  Conclusion: This `contact_reason` column seems to be MAR dependent on column `allegation` whose p-value is also 0.0. I also checked the column `rank_abbrev_incident`, which seems not to have relationship with the missingness of `contact_reason` in common sense. The p-value is 0.324, and we cannot reject the null hypothesis, this `rank_abbrev_incident` should have no effect on the missingness of `contact_reason`.

Hypothesis Test on column `allegation`

In [117]:

```
'''
Get the observed statistic from the original DF
Firstly, let's get the bar plot firstly
'''
obs = DF.pivot_table(index='allegation',
                     columns='isna_reason',
                     aggfunc='size').apply(lambda x: x / x.sum(), axis=0)

'''
Then we calculate the tvd use the pivot table above.
'''
obs_TVD = obs.diff(axis=1).abs().iloc[:, -1].sum() / 2

'''
Permutation Test
'''
TVDs = []
for _ in range(1000):
    DF['sampled'] = np.random.permutation(DF['isna_reason'])
    sample =  DF.pivot_table(index='allegation',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
    TVDs.append(sample.diff(axis=1).iloc[:, -1].abs().sum() / 2)

TVDs = pd.Series(TVDs)
(TVDs >= obs_TVD).mean()
```

Out[117]:

0.0

Hypothesis Test on column `rank_abbrev_incident`

In [106]:

```
DF['isna_reason'] = DF['contact_reason'].isna()
```

```
'''
Get the observed statistic from the original DF
Firstly, let's get the bar plot firstly
'''
obs = DF.pivot_table(index='rank_abbrev_incident',
                     columns='isna_reason',
                     aggfunc='size').apply(lambda x: x / x.sum(), axis=0)

'''
Then we calculate the tvd use the pivot table above.
'''
obs_TVD = obs.diff(axis=1).abs().iloc[:, -1].sum() / 2

'''
Permutation Test
'''
TVDs = []
for _ in range(1000):
    DF['sampled'] = np.random.permutation(DF['isna_reason'])
    sample =  DF.pivot_table(index='rank_abbrev_incident',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
    TVDs.append(sample.diff(axis=1).iloc[:, -1].abs().sum() / 2)

TVDs = pd.Series(TVDs)
(TVDs >= obs_TVD).mean()
```

Out[106]:

```
0.324
```

- **outcome_description: This missing type is hard to verify. I guess the missing type of precinct should be MAR, it should have some relationship with the column `contact_reason`. Please check the series below, which shows that the missingness should have be affected by `contact_reason` column. Of course it is also reasonable to say that those `contact_reason`'s cardinal numbers are large, so you can see most of the records missing `outcome_description` have the `contact_reason` attribute equal to something like 'PD suspected C/V of violation/crime - bldg' which is also frequently seen in the not missing records.**

  Since the proportion of null and non-null values are very different, I cannot verify my hypothesis here.

In [293]:

```
DF[DF['outcome_description'].isna()]['contact_reason'].value_counts()
```

Out[293]:

```
PD suspected C/V of violation/crime - bldg      17
EDP aided case                                  11
PD suspected C/V of violation/crime - street    11
PD telephones CV                                 4
Moving violation                                 3
Report of other crime                            2
Report-domestic dispute                          2
Other                                            2
C/V at PCT to retrieve property                  1
Report-dispute                                   1
C/V at PCT to obtain information                 1
Name: contact_reason, dtype: int64
```

- **precinct: This missing type is hard to verify. I guess the missing type of `precinct` should be MAR, it should have some relationship with the column `command_at_incident`. If the command of the officer is not related to a specified precinct (e.g. Warrant Section, or Detective Squad).**

  However, the proportion of null and non-null values are very different. I cannot verify my hypothesis here.

- **allegation: This missing type can be considered as MCAR, since only one record is missing. Also, this record contains very little information (lots of other attributes are missing). In addition, this complaint is received** contains very little information (lots of other attributes are missing). In addition, this complaint is received and closed in the same month, which is rare to see in other records (about 4%), we can consider that this record is generated by accident, and with many missing information as a result. It is fine to drop this record.

In [119]:

```
DF[DF['allegation'].isna()].head()
DF = DF.dropna(subset=['allegation'])
DF['allegation'].isna().sum()
```

Out[119]:

0

Hypothesis Test

We conduct a further study on my bivariate analysis where I check the relationship between the disposition of the officers and the outcome of the complainants and calculate the TVD (0.133). I will conduct a hypothesis test here to confirm my guess.

Here comes the null hypothesis and alternative hypothesis:
Null Hypothesis: **Arrested Complainants/ Nonarrested Complainants' complaints are dealt equally (same distribution).**
Alternative Hypothesis: **Arrested Complainants / Nonarrested Complanants' complaints are dealt unequally (different distribution).**
Test Statistc: **Total-Variation-Distance (TVD)**
Significance Level: **0.01**

Conclusion: **The p_value is 0.0, and we can reject the null hypothesis. That is, we can conclude that whether the complaint's is guity and arrested do affect the punishiment of the officers. You can see that the officers are more likely to be exonerated if the complainants are guilty themselves.**

In [130]:

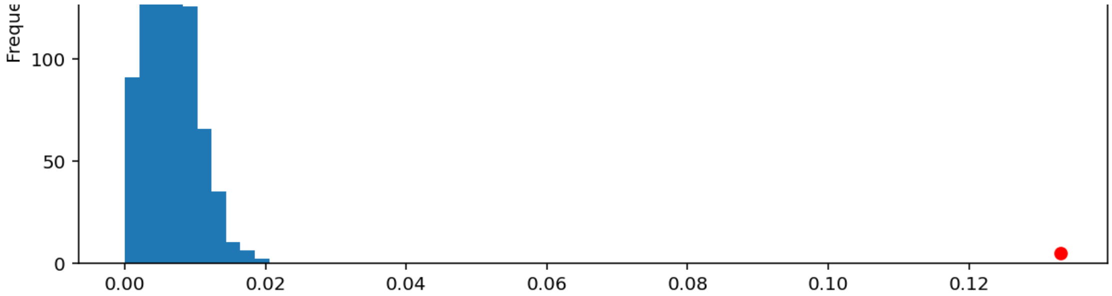
```
DF['sampled'] = np.random.permutation(DF['outcome_description'])
    sample =  DF.pivot_table(index='board_disposition',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
```

In [137]:

```
'''
Since we have already got the obs_TVD which equals to 0.133, we do the permutation test
directly here, and try to get the p-value.
'''
obs_TVD = 0.133
TVDs = []
for _ in range(1000):
    DF['sampled'] = np.random.permutation(DF['outcome_description'])
    sample =  DF.pivot_table(index='board_disposition',
                             columns='sampled',
                             aggfunc='size').apply(lambda x: x / x.sum(), axis=0)
    TVDs.append(sample.diff(axis=1).iloc[:, -1].abs().sum() / 2)
pd.Series(TVDs).plot(kind='hist', bins=10, figsize=(10,5))
plt.scatter(obs_TVD, 5, color='red', s=40, zorder=10)
```

Out[137]:

<matplotlib.collections.PathCollection at 0x1c417448>



In [140]:

```
p_value = (pd.Series(TVDs) >= obs_TVD).mean()
p_value
```

Out[140]:

```
0.0
```