stocks

February 4, 2023

1 Introduction to this Report

This is a project assignment from DSC 80 at UCSD instructed by Professor Justin Eldridge. In collaboration with Zhengyun Nie(znie@ucsd.edu), as a group, we analyzed the stock trades of the US House of Representatives. In this report, we cleaned the data, explored the data through univariate and bivariate anlysis as well as aggregation, assessed the missingness of one of the columns, and conducted a hypothesis test on whether the trades are evenly distributed on each weekday.

2 Stock Trades by Members of the US House of Representatives

This project uses public data about the stock trades made by members of the US House of Representatives. This data is collected and maintained by Timothy Carambat as part of the House Stock Watcher project. The project describes itself as follows:

> With recent and ongoing investigations of incumbent congressional members being investigated for potentially violating the STOCK act. This website compiles this publicly available information in a format that is easier to digest then the original PDF source.
>
> Members of Congress must report periodic reports of their asset transactions. This website is purely for an informative purpose and aid in transparency.
>
> This site does not manipluate or censor any of the information from the original source. All data is transcribed by our community of contributors, which you can join for free by going to our transcription tool. Our moderation team takes great care in ensuring the accuracy of the information.
>
> This site is built and maintained by Timothy Carambat and supported with our contributors.

Some interesting questions to consider for this data set include:

- Is there a difference in stock trading behavior between political parties? For example:
  - does one party trade more often?
  - does one party make larger trades?
  - do the two parties invest in different stocks or sectors? For instance, do Democrats invest in Tesla more than Republicans?
- What congresspeople have made the most trades?
- What companies are most traded by congresspeople?
- Is there evidence of insider trading? For example, Boeing stock dropped sharply in February 2020. Were there a suspiciously-high number of sales of Boeing before the drop?

- When are stocks bought and sold? Is there a day of the week that is most common? Or a month of the year?

2.0.1 Getting the Data

The full data set of stock trade disclosures is available as a CSV or as JSON at https://housestockwatcher.com/api.

This data set does not, however, contain the political affiliation of the congresspeople. If you wish to investigate a question that relies on having this information, you'll need to find another dataset that contains it and perform a merge. *Hint*: Kaggle is a useful source of data sets.

2.0.2 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 "–." Do some exploration to find those values and convert them to null values.
  - You may also want to clean up the date columns to enable time-series exploration.
- Understand the data in ways relevant to your question using univariate and bivariate analysis of the data as well as aggregations.

2.0.3 Assessment of Missingness

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

2.0.4 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.

3 Summary of Findings

3.0.1 Introduction

This dataset records the stock transactions made by members of the US House of Representatives. - disclosure_year: the year when the transaction was disclosed. - disclosure_date: the date when the transaction was disclosed. - transaction_date: the date when the transaction was made. - owner: how the stock is owned by the representative(e.g. joint, self, or dependent). • ticker: the abbreviation of asset's company name. - asset_description: detailed description of the asset. • type: type of transaction(e.g. purchase, sale_full). - amount: a range of the amount of transaction. - representative: name of the representative. - district: district of the representative. - ptr_link: link of the source. - cap_gains_over_200_usd: whether the capital gain is greater than 200 USD. We want to intestigate into whether the transactions tend to make in a day of the week.

3.0.2 Cleaning and EDA

- Drop duplicate rows so that we will not overcount missingness or other sample size in later anylasis

- The original data set used '–' to indicate missingness, we change it to np.nan which makes us more convient to use certain functions.
- Change the time string in the dataframe to datetime object so that it is eaiser for us to find which days it is during one week.
- Add a year column represents the disclosure year of a stock which helps us find general pattern in differnt years.
- Fix typos in the year column to make it consistent with other values.
- Add a column to represent which day it is during the week in order to run our hypothesis tesing.
- Remove title of representitives and takes only the initial of middle names to avoid duplicates
- Split amount into two columns so that it has a lower bound and upper bound to help us to see more patterns.

3.0.3 Assessment of Missingness

We first calculated the proportion of missingness for each variable in the cleaned dataset. We noticed that column 'ticker' has about 8% of missingness, which is worth discovering. Looking into the dataset, we believe that the missingness of 'ticker' is NMAR because many rows with missing 'ticker' have descriptions that include the company name. There is no reason people don't want to report ticker information if they include them in description, so ticker itself should not affect missingness. We then looks into two columns: 'cap_gains_over_200_usd' and 'transaction_weekday'.

- The Relationship between 'Ticker's Missingness and Values of 'cap_gains_over_200_usd' **Null Hypothesis**: The missingness of 'ticker' column is independent of values of 'cap_gains_over_200_usd' column.

  Alternative Hypothesis: The missingness of 'ticker' column depends on values of 'cap_gains_over_200_usd' column.

  Test Statistic: Because values of 'cap_gains_over_200_usd' are categorical and our alternative hypothesis does not have direction, we choose TVD as our test statistic.

  Conclusion: Since p-value is 0, we can reject the null hypothesis. Thus, missingness of 'ticker' is MAR and depends on 'cap_gains_over_200_usd'.

- The Relationship between 'Ticker's Missingness and Values of 'transaction_weekday' **Null Hypothesis**: The missingness of 'ticker' column is independent of values of 'transaction_weekday' column.

  Alternative Hypothesis: The missingness of 'ticker' column depends on values of 'transaction_weekday' column.

  Test Statistic: Because values of 'transaction_weekday' are categorical and our alternative hypothesis does not have direction, we choose TVD as our test statistic.

  Conclusion: We have p-value of 0.056. With significant level of 5%, since p-value is greater than 0.05, we fail to reject the null hypothesis. Therefore, missingness of ticker is independent of transaction_weekday.

3.0.4 Hypothesis Test

First, we want to study whether all the transactions are evenly spread over the workdays(i.e from Monday to Friday). Our null hypothesis is that all the transactions are evenly occurred on each of the workdays with probability of 0.2. The correspondnig alternative hypothesis would be that there is one particular day such that more transactions are likely to happen. To test this, we first calculate the proportion of transactions on each day we observed and calculate the tvd with our model that has proportion of 0.2 on each day in order to see if they come from the same distribution. We set the significant level as 0.05 as it is a stanard one. The p value we get is around 0.012 and hence we rejected our null hypothesis and conclude that all the transactions are not evenly occured on each of the workdays.

4 Code

[1]:
```
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
```

4.0.1 Cleaning and EDA

Clean the data

[2]:
```
# import data from .csv
raw_stocks = pd.read_csv(os.path.join('data', 'all_transactions.csv'))
raw_stocks.head() # part of the raw dataset
```

[2]:

	disclosure_year	disclosure_date	transaction_date	owner	ticker	asset_description	type	amount	representative	district
0	2021	10/04/2021	2021-09-27	joint	BP	BP plc	purchase	$1,001 - $15,000	Hon. Virginia Foxx	NC05
1	2021	10/04/2021	2021-09-13	joint	XOM	Exxon Mobil Corporation	purchase	$1,001 - $15,000	Hon. Virginia Foxx	NC05
2	2021	10/04/2021	2021-09-10	joint	ILPT	Industrial Logistics Properties Trust - Common…	purchase	$15,001 - $50,000	Hon. Virginia Foxx	NC05
3	2021	10/04/2021	2021-09-28	joint	PM	Phillip Morris International Inc	purchase	$15,001 - $50,000	Hon. Virginia Foxx	NC05
4	2021	10/04/2021	2021-09-17	self	BLK	BlackRock Inc	sale_partial			

```
4   $1,001 - $15,000  Hon. Alan S. Lowenthal     CA47

                                          ptr_link  cap_gains_over_200_usd
0  https://disclosures-clerk.house.gov/public_dis…                   False
1  https://disclosures-clerk.house.gov/public_dis…                   False
2  https://disclosures-clerk.house.gov/public_dis…                   False
3  https://disclosures-clerk.house.gov/public_dis…                   False
4  https://disclosures-clerk.house.gov/public_dis…                   False
```

[3]:
```
# drop duplicate rows
raw_stocks = raw_stocks.drop_duplicates()
```

[4]:
```
# turn '--' into NaN
raw_stocks['ticker'] = raw_stocks['ticker'].replace('--', np.NaN)
raw_stocks['owner'] = raw_stocks['owner'].replace('--', np.NaN)
```

[5]:
```
# adjust date to DateTime
raw_stocks['disclosure_date'] = pd.to_datetime(raw_stocks['disclosure_date'])
```

[6]:
```
# keep disclosure year consistent with disclosure_date
raw_stocks['disclosure_year'] = raw_stocks['disclosure_date'].dt.year
```

[7]:
```
# Wrong data: '0009-06-09' likely to be '2021-06-09'
raw_stocks['transaction_date'] = pd.to_datetime(raw_stocks['transaction_date'].
 ↪replace({'0009-06-09': '2021-06-09',\
                                                  '20222-08-09':␣
 ↪'2022-08-09','0021-08-02': '2021-08-02', \
                                                  '20222-07-18':␣
 ↪'2022-07-18','20221-11-18': '2021-11-18',\
                                                  '0021-06-22':␣
 ↪'2021-06-22','0201-06-22':'2021-06-22'}))
```

[8]:
```
# Obtain day of week for each transaction
raw_stocks['transaction_weekday'] = raw_stocks['transaction_date'].dt.weekday
```

[9]:
```
# Removes title of representitives and takes only the initial of middle names␣
 ↪to avoid duplicates
def deal_name(name):
    temp = name.split()
    full_name = temp[1:]
    if len(full_name) > 2:
        for i in range(1, len(full_name)-1):
            full_name[i] = full_name[i][0].upper()+'.'
    return ' '.join(full_name)
```

[10]:
```

```

```
# Clean name
raw_stocks['full_name'] = raw_stocks['representative'].apply(lambda x:␣
↪deal_name(x))
```

[11]:
```
# split amount into two columns
temp = raw_stocks['amount'][raw_stocks['amount'].str.contains('-')].str.
↪split('-')
raw_stocks['amount_lower_range'] = temp.apply(lambda x: -100 if x[0] == '' else␣
↪x[0].strip()[1:].replace(',', '')).astype(int).replace(-100, np.NaN)
raw_stocks['amount_upper_range'] = temp.apply(lambda x: -100 if x[-1] == ''␣
↪else x[-1].strip()[1:].replace(',', '')).astype(int).replace(-100, np.NaN)
temp = raw_stocks['amount'][~raw_stocks['amount'].str.contains('-')].str.
↪split('+').apply(lambda x: -100 if x[0] == '' else x[0].strip()[1:].
↪replace(',', ''))
raw_stocks['amount_lower_range'].loc[temp.index] = temp.astype(int)
stocks = raw_stocks.copy()
stocks.head() # head of cleaned dataset
```

```
/Users/maowanting/opt/anaconda3/lib/python3.9/site-
packages/pandas/core/indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

[11]:

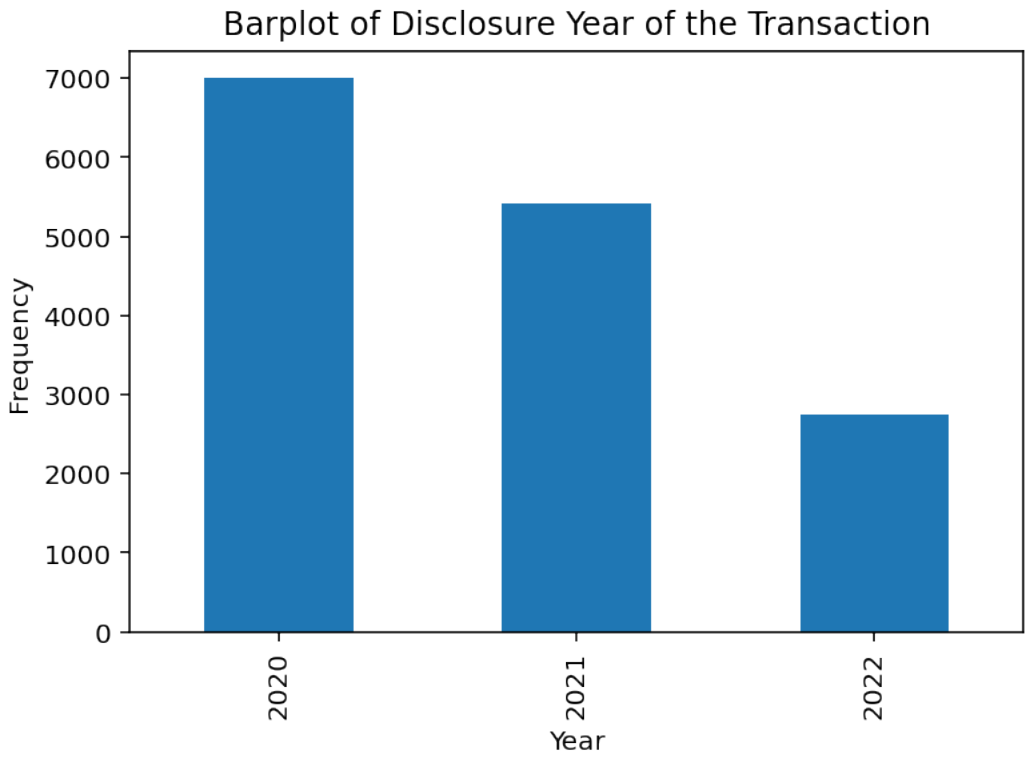
	disclosure_year	disclosure_date	transaction_date	owner	ticker	asset_description	type	amount	representative	district
0	2021	2021-10-04	2021-09-27	joint	BP	BP plc	purchase	$1,001 - $15,000	Hon. Virginia Foxx	NC05
1	2021	2021-10-04	2021-09-13	joint	XOM	Exxon Mobil Corporation	purchase	$1,001 - $15,000	Hon. Virginia Foxx	NC05
2	2021	2021-10-04	2021-09-10	joint	ILPT	Industrial Logistics Properties Trust - Common…	purchase	$15,001 - $50,000	Hon. Virginia Foxx	NC05
3	2021	2021-10-04	2021-09-28	joint	PM	Phillip Morris International Inc	purchase	$15,001 - $50,000	Hon. Virginia Foxx	NC05
4	2021	2021-10-04	2021-09-17	self	BLK	BlackRock Inc	sale_partial	$1,001 - $15,000	Hon. Alan S. Lowenthal	CA47

```
                                              ptr_link  cap_gains_over_200_usd  \
0  https://disclosures-clerk.house.gov/public_dis…                      False
1  https://disclosures-clerk.house.gov/public_dis…                      False
2  https://disclosures-clerk.house.gov/public_dis…                      False
3  https://disclosures-clerk.house.gov/public_dis…                      False
4  https://disclosures-clerk.house.gov/public_dis…                      False

   transaction_weekday          full_name  amount_lower_range  \
0                    0      Virginia Foxx              1001.0
1                    0      Virginia Foxx              1001.0
2                    4      Virginia Foxx             15001.0
3                    1      Virginia Foxx             15001.0
4                    4  Alan S. Lowenthal              1001.0

   amount_upper_range
0             15000.0
1             15000.0
2             50000.0
3             50000.0
4             15000.0
```

Univariate Analysis This plot shows the number of disclosure in each year. From the plot, most disclosures were taken place in 2020.

[12]:
```
# disclosure_year
stocks['disclosure_year'].value_counts().plot(kind = 'bar')
plt.title('Barplot of Disclosure Year of the Transaction')
plt.xlabel('Year')
plt.ylabel('Frequency')
```

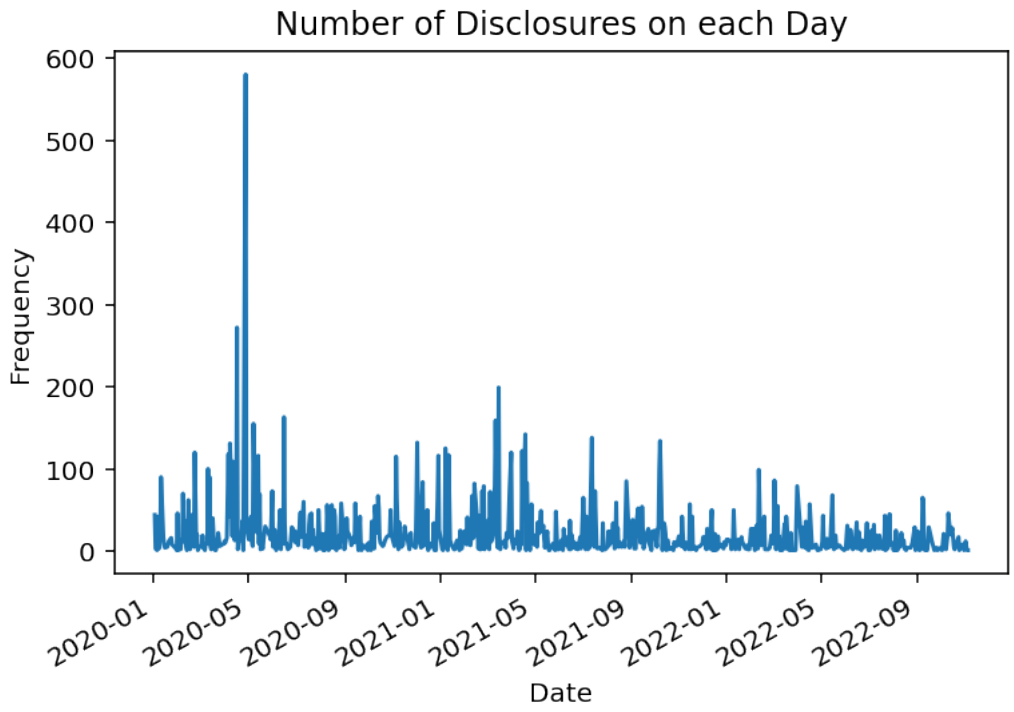
[12]:
```
Text(0, 0.5, 'Frequency')
```



Below shows the number of disclosures on each day. The greatest number of disclosures was on 2020-04-27, where 581 cases are disclosed.

[13]:
```
# disclosure_date
stocks['disclosure_date'].value_counts().plot(kind = 'line')
plt.title('Number of Disclosures on each Day')
plt.xlabel('Date')
plt.ylabel('Frequency')
```

[13]:
```
Text(0, 0.5, 'Frequency')
```



[14]:
```
stocks['disclosure_date'].value_counts()
```

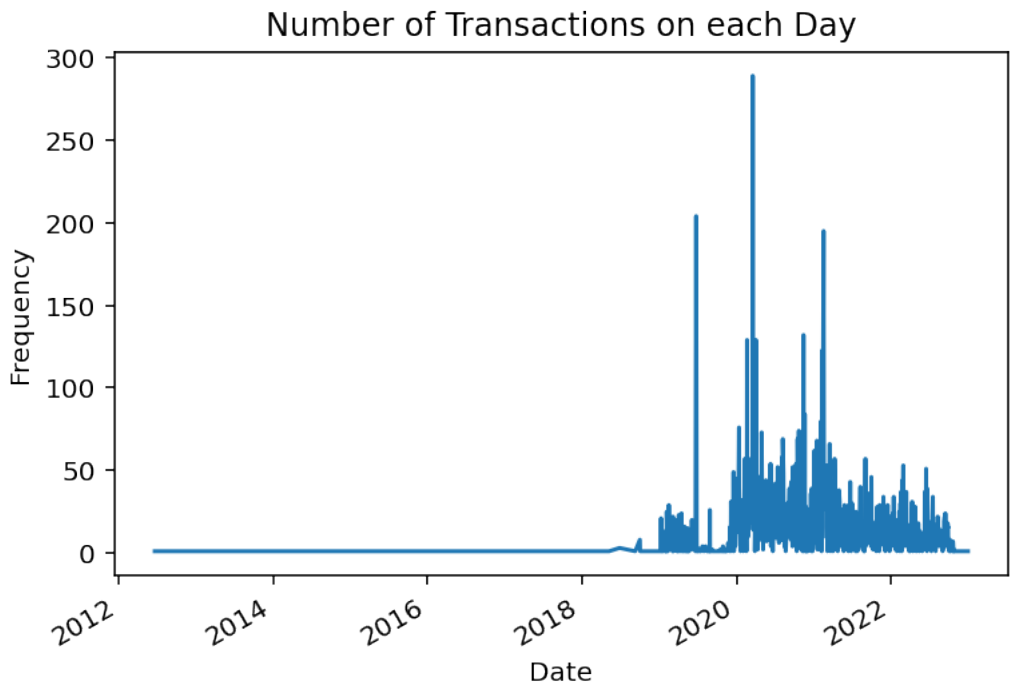
[14]:
```
2020-04-27    580
2020-04-16    272
2021-03-16    199
2020-06-15    163
2021-03-12    159
              ...
2022-04-28      1
2021-11-23      1
2022-08-30      1
2020-03-06      1
2022-09-27      1
Name: disclosure_date, Length: 732, dtype: int64
```

Below shows the number of disclosures on each day. In general, not many transactions appear before 2018, which might due to the fact that transcations taken place before 2018 is not required to report, so those transactions were not recorded in this dataset. Several notable peaks are on 2020-03-18, 2019-06-24, and 2021-02-16.

[15]:
```
# transaction_date
stocks['transaction_date'].value_counts().plot(kind = 'line')
plt.title('Number of Transactions on each Day')
plt.xlabel('Date')
```

```
plt.ylabel('Frequency')
```

[15]: `Text(0, 0.5, 'Frequency')`

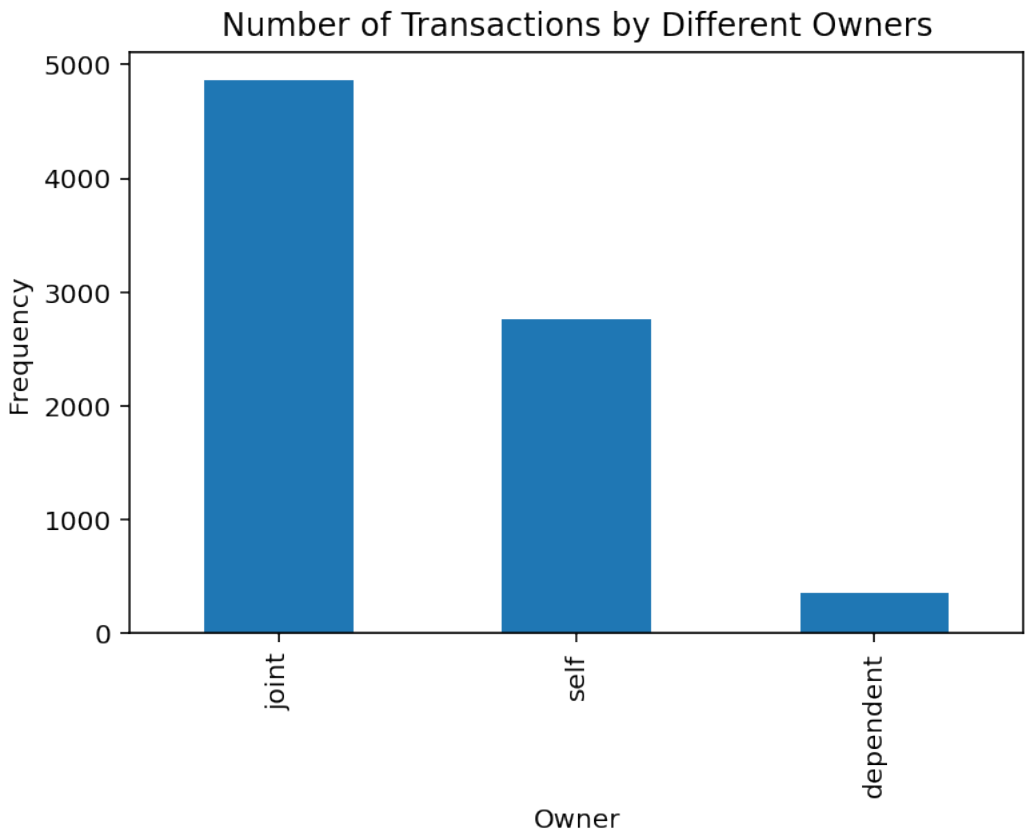


[16]:
```
stocks['transaction_date'].value_counts()
```

[16]:
```
2020-03-18    289
2019-06-24    204
2021-02-16    195
2020-11-13    132
2020-02-20    129
              ...
2019-05-08      1
2019-05-09      1
2019-03-08      1
2019-01-30      1
2020-03-28      1
Name: transaction_date, Length: 927, dtype: int64
```

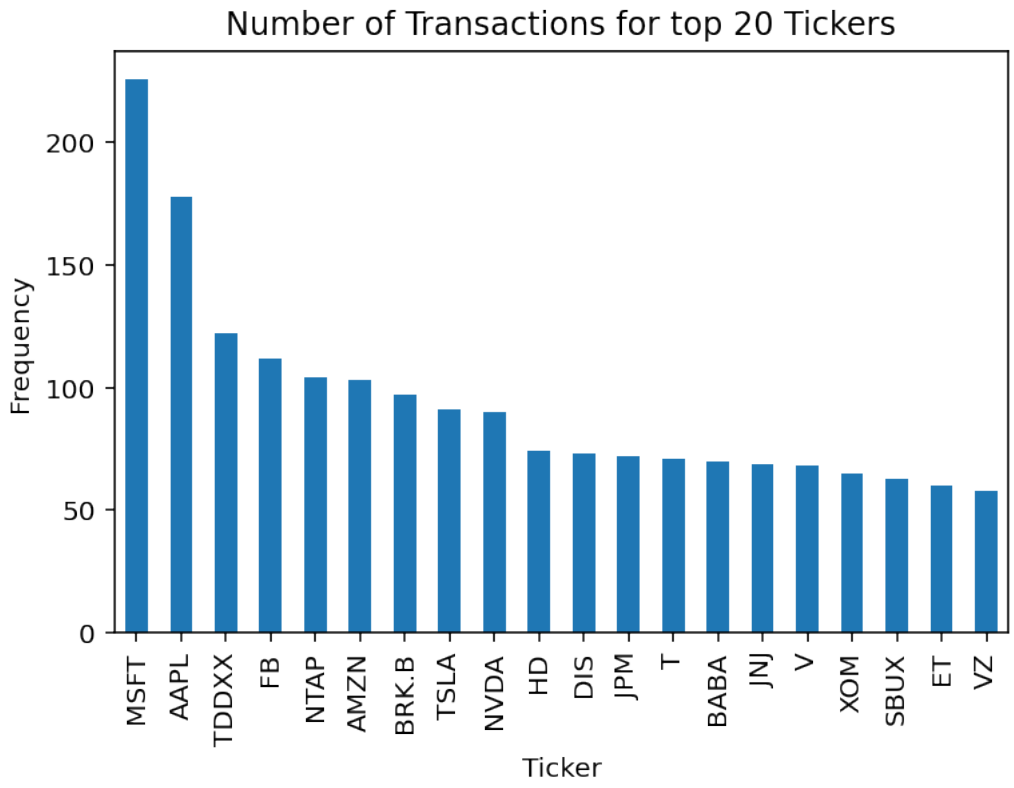
[17]:
```
# owner
stocks['owner'].value_counts().plot(kind = 'bar')
plt.title('Number of Transactions by Different Owners')
plt.xlabel('Owner')
plt.ylabel('Frequency')
```

[17]: Text(0, 0.5, 'Frequency')



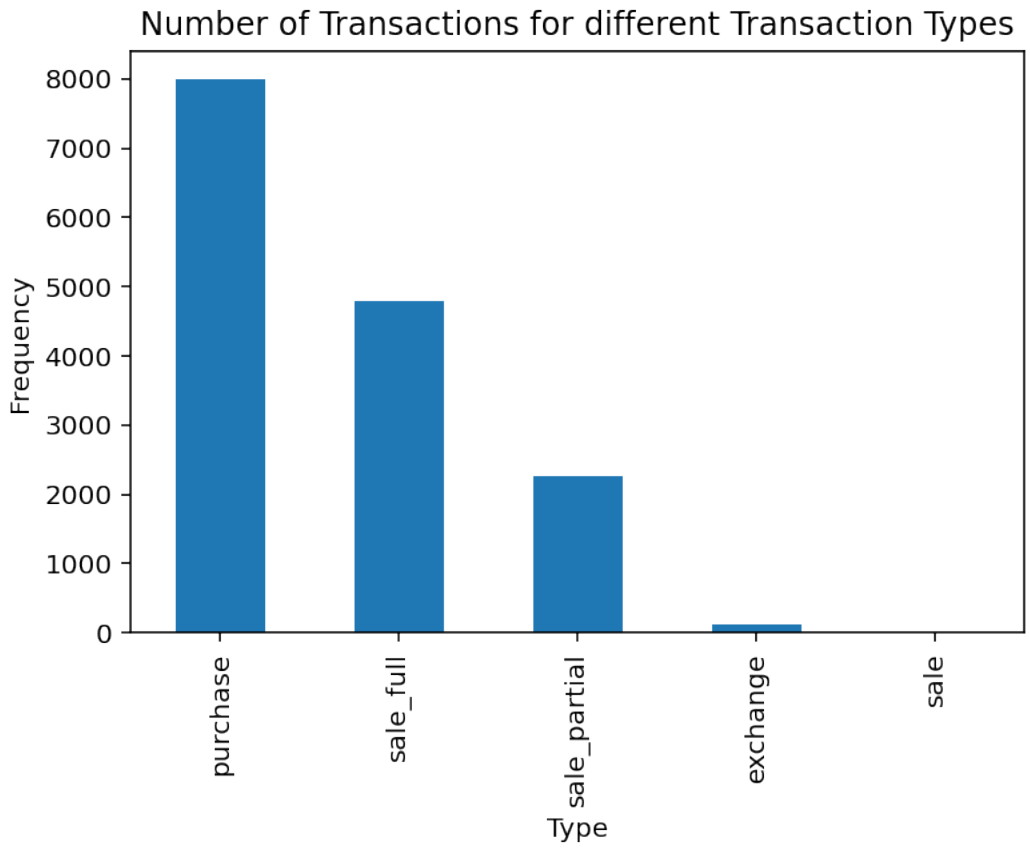
[18]:
```
# ticker
stocks['ticker'].value_counts().iloc[:20].plot(kind = 'bar')
plt.title('Number of Transactions for top 20 Tickers')
plt.xlabel('Ticker')
plt.ylabel('Frequency')
```

[18]: Text(0, 0.5, 'Frequency')



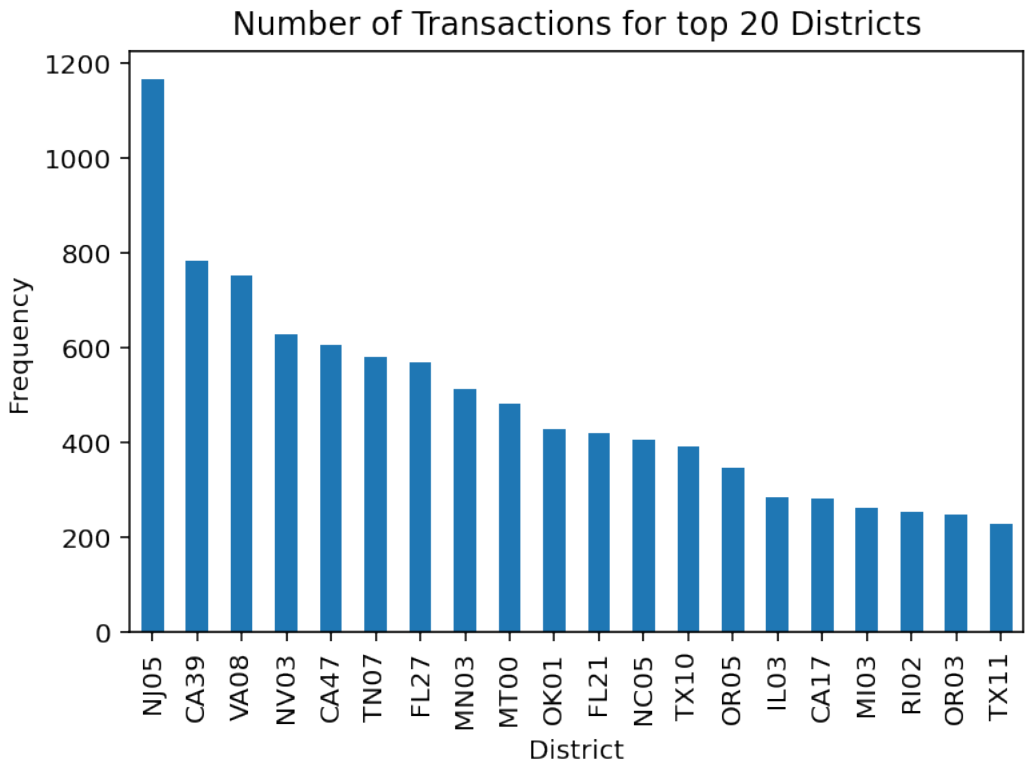
[19]:
```
# type
stocks['type'].value_counts().iloc[:20].plot(kind = 'bar')
plt.title('Number of Transactions for different Transaction Types')
plt.xlabel('Type')
plt.ylabel('Frequency')
```

[19]:
```
Text(0, 0.5, 'Frequency')
```



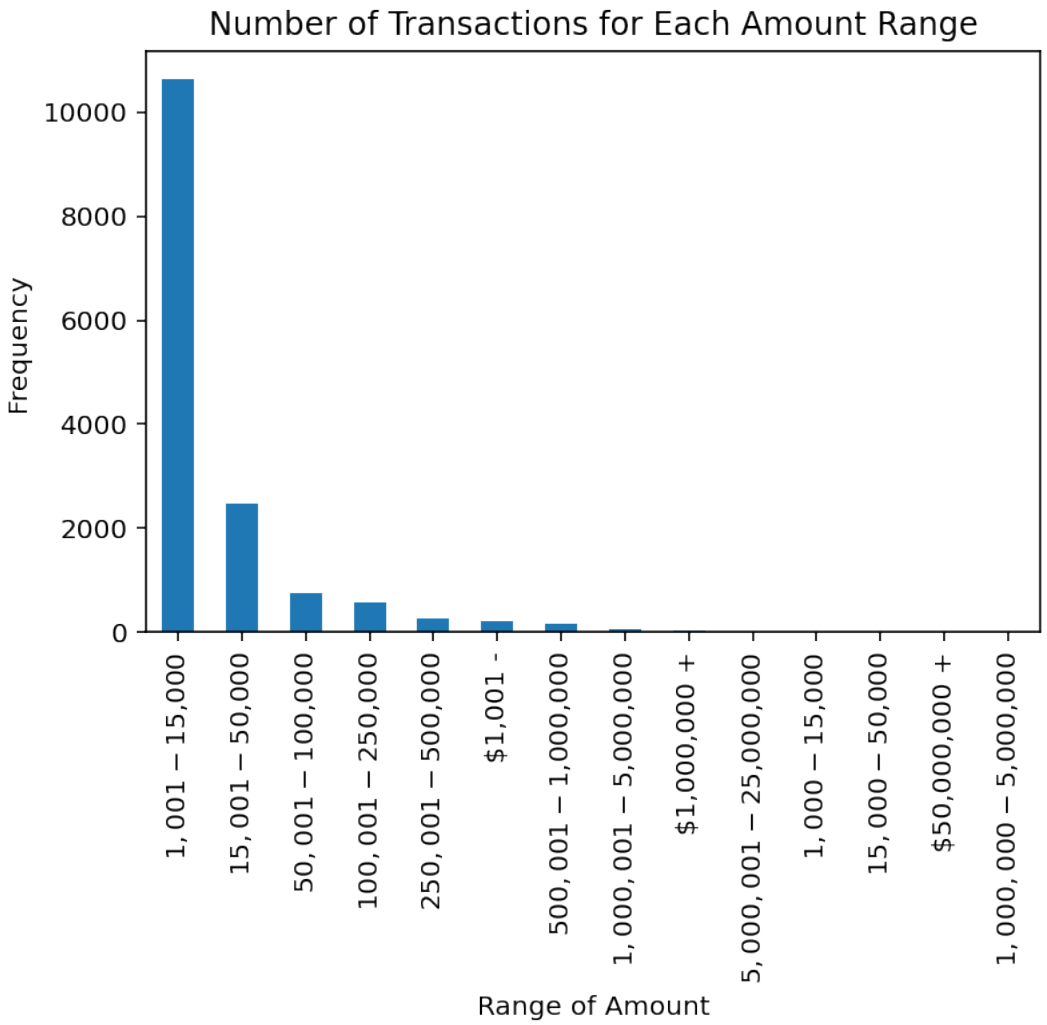
[20]:
```
# district
stocks['district'].value_counts()[:20].plot(kind = 'bar')
plt.title('Number of Transactions for top 20 Districts')
plt.xlabel('District')
plt.ylabel('Frequency')
```

[20]:
```
Text(0, 0.5, 'Frequency')
```



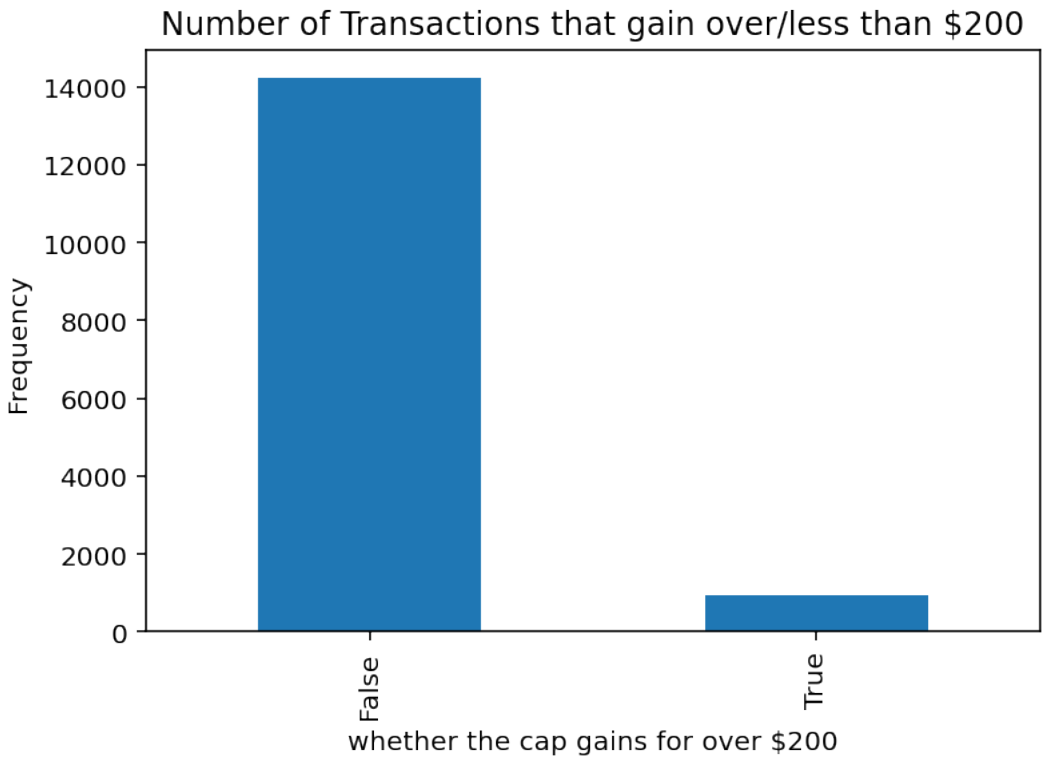
[21]:
```
# amount
stocks['amount'].value_counts().plot(kind = 'bar')
plt.title('Number of Transactions for Each Amount Range')
plt.xlabel('Range of Amount')
plt.ylabel('Frequency')
```

[21]: Text(0, 0.5, 'Frequency')



[22]:
```
# whether the cap gains for over $200
stocks['cap_gains_over_200_usd'].value_counts().plot(kind = 'bar')
plt.title('Number of Transactions that gain over/less than $200')
plt.xlabel('whether the cap gains for over $200')
plt.ylabel('Frequency')
```

[22]:
```
Text(0, 0.5, 'Frequency')
```



Bivariate Analysis

[23]:
```
ticker_gains = stocks.pivot_table(index = 'ticker',columns =␣
↪'cap_gains_over_200_usd', aggfunc = 'size').fillna(0.0)
ticker_gains
```

[23]:

cap_gains_over_200_usd	False	True
ticker		
35G.SG	2.0	0.0
7XY	1.0	0.0
A	1.0	2.0
AA	10.0	0.0
AABV	1.0	0.0
…	…	…
ZM	13.0	0.0
ZNGA	3.0	0.0
ZOOM	1.0	1.0
ZTS	13.0	1.0
ZUO	3.0	1.0

[2218 rows x 2 columns]

[24]:
```
amount_gains = stocks.pivot_table(index = 'amount',columns =␣
↪'cap_gains_over_200_usd', aggfunc = 'size').fillna(0.0)
amount_gains
```

[24]:

cap_gains_over_200_usd	False	True
amount		
$1,000 - $15,000	4.0	0.0
$1,000,000 +	29.0	1.0
$1,000,000 - $5,000,000	1.0	0.0
$1,000,001 - $5,000,000	44.0	2.0
$1,001 -	189.0	17.0
$1,001 - $15,000	10013.0	625.0
$100,001 - $250,000	537.0	36.0
$15,000 - $50,000	3.0	0.0
$15,001 - $50,000	2295.0	186.0
$250,001 - $500,000	245.0	15.0
$5,000,001 - $25,000,000	8.0	1.0
$50,000,000 +	1.0	0.0
$50,001 - $100,000	706.0	51.0
$500,001 - $1,000,000	153.0	6.0

Interesting aggregation

[25]:
```
### information between different representative
stocks.groupby('full_name').size()
```

[25]:
```
full_name
Abigail Spanberger      3
Adam B. Schiff          4
Adam Kinzinger         24
Adrian Smith            1
Alan S. Lowenthal     606
                      …
W. G. Steube            2
William R. Keating     80
William R. Timmons     18
Wm. L. Clay             2
Zoe Lofgren           201
Length: 180, dtype: int64
```

[26]:
```
### information between transaction amounts and district
stocks.groupby(['amount','district']).size()
```

[26]:
```
amount                district
$1,000 - $15,000      FL15        4
$1,000,000 +          CA06       10
                      CA17        2
                      CA52       16
```

```
                        IA01         1
                                    ..
$500,001 - $1,000,000   TX03         2
                        TX10         9
                        VA08         1
                        WA01        35
                        WA08         2
Length: 483, dtype: int64
```

4.0.2 Assessment of Missingness

[27]:
```
for col in stocks.columns:
    print('Proportion of Missingness for '+col+': '+str(sum(stocks[col].isna())/
stocks.shape[0]))
```

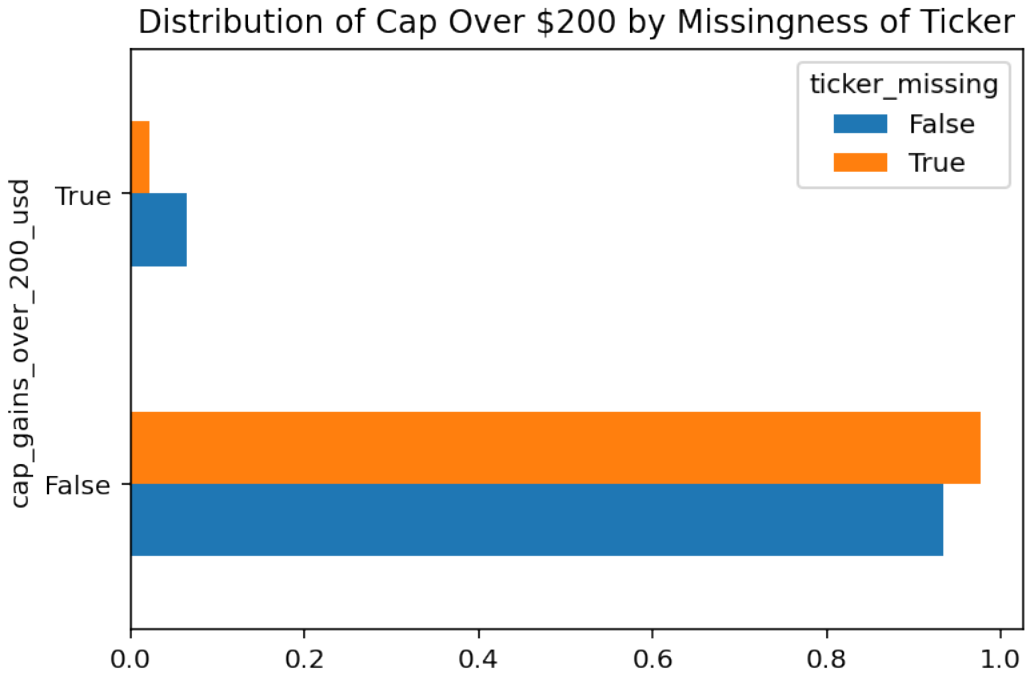
```
Proportion of Missingness for disclosure_year: 0.0
Proportion of Missingness for disclosure_date: 0.0
Proportion of Missingness for transaction_date: 0.0
Proportion of Missingness for owner: 0.47356276371308015
Proportion of Missingness for ticker: 0.08438818565400844
Proportion of Missingness for asset_description: 0.00026371308016877635
Proportion of Missingness for type: 0.0
Proportion of Missingness for amount: 0.0
Proportion of Missingness for representative: 0.0
Proportion of Missingness for district: 0.0
Proportion of Missingness for ptr_link: 0.0
Proportion of Missingness for cap_gains_over_200_usd: 0.0
Proportion of Missingness for transaction_weekday: 0.0
Proportion of Missingness for full_name: 0.0
Proportion of Missingness for amount_lower_range: 0.0
Proportion of Missingness for amount_upper_range: 0.015625
```

It's notable that 'ticker' column has about 8% of missingness which is relatively high. We believe the missingness in 'ticker' is not due to NMAR because many missing rows has description that include the company name, so ticker itself should not affect missingness.

The Relationship between the Missingness of 'ticker' Column and Values of 'cap_gains_over_200_usd' column

[28]:
```
df = stocks.copy()
df['ticker_missing'] = df['ticker'].isna()
cap_200_dist = (
    df.pivot_table(index = 'cap_gains_over_200_usd', columns =␣
↪'ticker_missing',values=None, aggfunc = 'size')
    .fillna(0)
    .apply(lambda x: x / x.sum())
)
```

```
cap_200_dist.plot(kind='barh', title='Distribution of Cap Over $200 by␣
↪Missingness of Ticker');
```



Further, we oberseved that when 'cap_gains_over_200_usd' is true, less proportion of ticker is missing. Thus, we tend to test whether the missingness of 'ticker' column relates to values of 'cap_gains_over_200_usd'(i.e. the missingness of 'ticker' is MAR due to 'cap_gains_over_200_usd').

Null Hypothesis: The missingness of 'ticker' column is independent of values of 'cap_gains_over_200_usd' column.

Alternative Hypothesis: The missingness of 'ticker' column depends on values of 'cap_gains_over_200_usd' column.

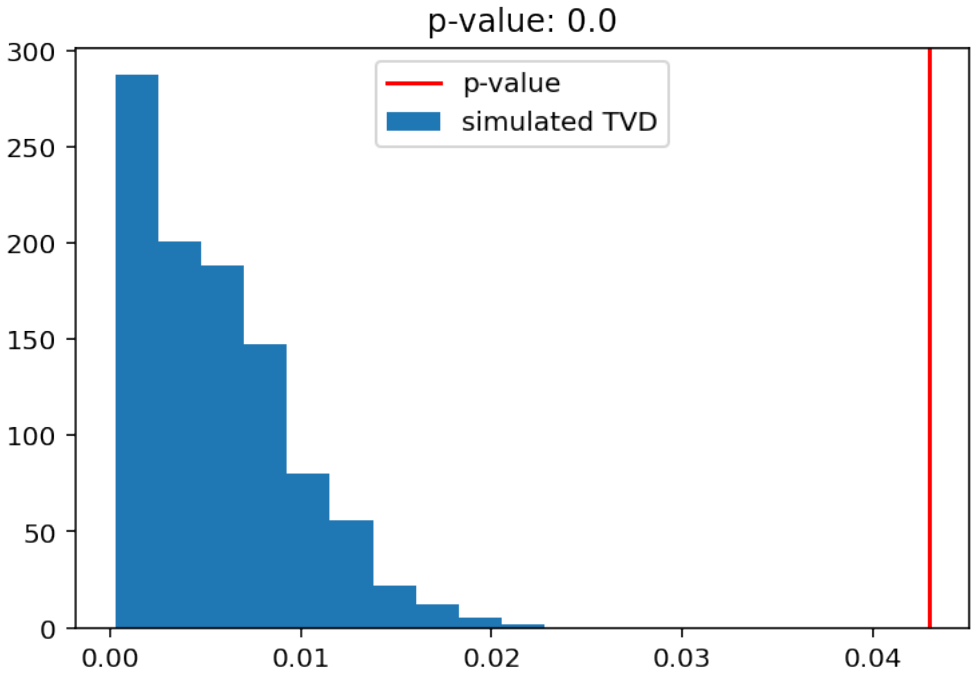
Test Statistic: Because values of 'cap_gains_over_200_usd' are categorical and our alternative hypothesis does not have direction, we choose TVD as our test statistic.

[29]:
```
tvds = []
# Compute observational test statistics
obs_tvd = cap_200_dist.diff(axis=1).iloc[:, -1].abs().sum() / 2

# Simulation
for _ in range(10**3):
    # shuffle 'cap_gains_over_200_usd' column
    df['shuffle_cap_over_200'] = np.random.
↪permutation(df['cap_gains_over_200_usd'])
```

```
    sim_cap_200_dist = (
        df.pivot_table(index = 'shuffle_cap_over_200', columns =␣
↪'ticker_missing',values=None, aggfunc = 'size')
        .fillna(0)
        .apply(lambda x: x / x.sum())
    )
    # Compute test statistic
    tvds.append(sim_cap_200_dist.diff(axis=1).iloc[:, -1].abs().sum() / 2)
```

[30]:
```
plt.hist(tvds, label = 'simulated TVD')
plt.axvline(x = obs_tvd, color = 'red', label = 'p-value')
plt.title('p-value: ' +str(np.mean(tvds>=obs_tvd)))
plt.legend()
```

[30]: <matplotlib.legend.Legend at 0x7fb8628ef0d0>



Conclusion: Since p-value is 0, we can reject the null hypothesis. Thus, missingness of 'ticker' is MAR and depends on 'cap_gains_over_200_usd'.

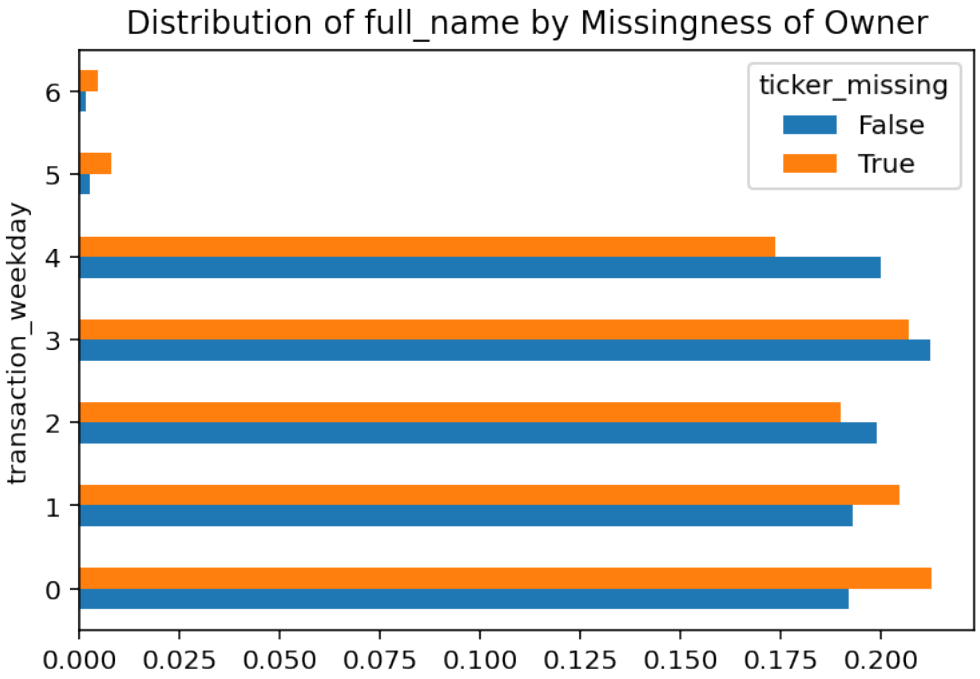
The Relationship between the Missingness of 'ticker' and Values of 'transaction_weekday'

Null Hypothesis: The missingness of 'ticker' column is independent of values of 'transaction_weekday' column.

Alternative Hypothesis: The missingness of 'ticker' column depends on values of 'transaction_weekday' column.

Test Statistic: Because values of 'transaction_weekday' are categorical and our alternative hypothesis does not have direction, we choose TVD as our test statistic.

[31]:
```
weekday_dist = (
    df.pivot_table(index = 'transaction_weekday', columns =␣
↪'ticker_missing',values=None, aggfunc = 'size')
    .fillna(0)
    .apply(lambda x: x / x.sum())
)
weekday_dist[:20].plot(kind='barh', title='Distribution of full_name by␣
↪Missingness of Owner');
```

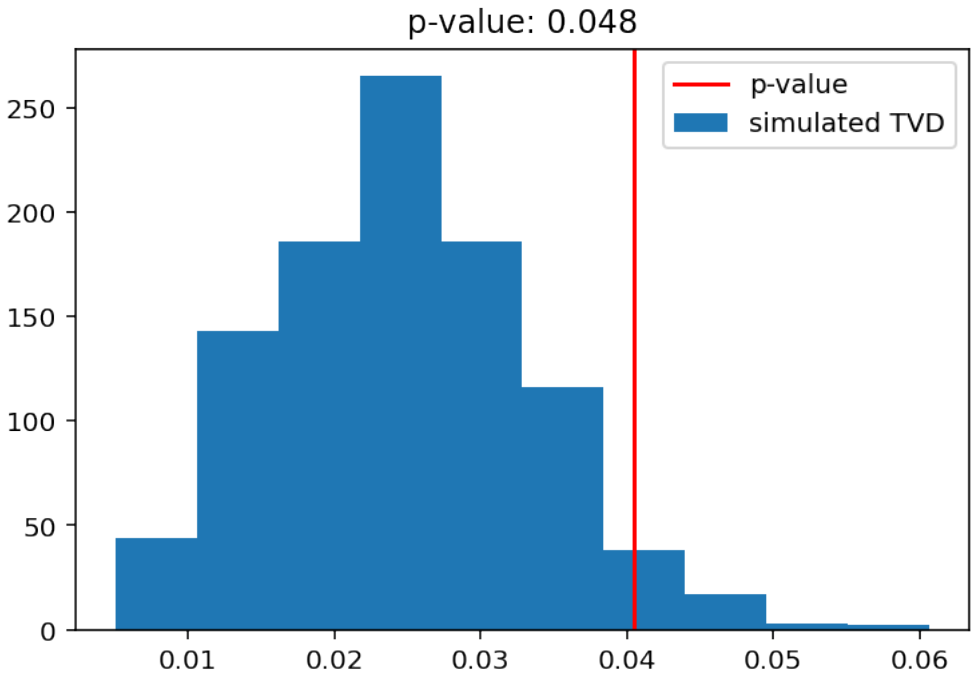


[32]:
```
tvds = []
# Compute observational test statistics
obs_tvd = weekday_dist.diff(axis=1).iloc[:, -1].abs().sum() / 2

# Simulation
for _ in range(10**3):
    # shuffle 'transaction_weekday' column
    df['shuffle_weekday'] = np.random.permutation(df['transaction_weekday'])
    sim_dist = (
```

```
        df.pivot_table(index = 'shuffle_weekday', columns =␣
↪'ticker_missing',values=None, aggfunc = 'size')
        .fillna(0)
        .apply(lambda x: x / x.sum())
    )
    # Compute test statistic
    tvds.append(sim_dist.diff(axis=1).iloc[:, -1].abs().sum() / 2)
```

[33]:
```
plt.hist(tvds, label = 'simulated TVD')
plt.axvline(x = obs_tvd, color = 'red', label = 'p-value')
plt.title('p-value: '+str(np.mean(tvds >= obs_tvd)))
plt.legend()
```

[33]:
```
<matplotlib.legend.Legend at 0x7fb862b07ee0>
```



Conclusion: With significant level of 5%, since p-value is less than 0.05, we reject the null hypothesis. Therefore, missingness of ticker is dependent on transaction_weekday.

4.0.3 Hypothesis Test / Permutation Test

We want to investigate that if transcations are usually occured particularly in one of the workdays(i.e one day in Monday through Friday).

4.0.4 Null Hypothesis

Our null Hypothesis is that all the transaction are occured evenly through out the weekdays as under this situation the probability of one transaction happened on one of the workday is roughly $\frac{1}{5}$.

4.0.5 Alternative Hypothesis

Our altenative Hypotheis is that there is one special day during the workday that has substantial transactions.

4.0.6 Test-Statistics

We will use TVD as our test-stastics as theoretically the porportion of transaction occured on each day is 0.2. We will use the sample we observed and genereated to compare to this model to see whether they come from the same distribution. We will also set the significance level to be 0.05 as it is somewhat standard.

[34]:
```
stocks['day_of_week'] = pd.to_datetime(stocks['transaction_date']).
 ↪transform(lambda x: x.day_of_week)
```

[35]:
```
## get the number of transaction that traded during workdays
workday_transaction = stocks.groupby(['day_of_week']).size()[:5].sum()
workday_transaction
```

[35]: 15093

[36]:
```
## get our observed data
obs = ((stocks.groupby('day_of_week').size()[:5] / workday_transaction) - 0.2).
 ↪abs().sum() / 2
obs
```

[36]: 0.01281388723249191

[37]:
```
## randomly built samples from our model and record TVD and compute p value
result = []
for i in range(2000):
    sample = np.random.choice([0,1,2,3,4], size = workday_transaction, p = [0.
 ↪2,0.2,0.2,0.2,0.2])
    df = pd.DataFrame(sample, columns = ['day_of_week'])
    result.append((df.groupby('day_of_week').size() / workday_transaction - 0.
 ↪2).abs().sum() / 2)
```

[38]:
```
(result >= obs).mean()
```

[38]: 0.01

Notice that the p value is only around 0.01, which is way lower than the p-value we set. Therefore, we should reject our null hypothesis that all the transaction are evenly occurred on each of the workday. In fact, we notice that slighly more transactions were happened on Thursday.