# Snapchat Political Ads Analysis

May 14, 2020

## 1 Summary of Findings

### 1.1 Introduction

This investigation uses two datasets, PoliticalAds2018.csv and PoliticalAds2019.csv. Both of these datasets come from Snap Inc. and contain information about every political, issue related, and advocacy ad shown on Snapchat in 2018 and 2019, respectively. The entries of the datasets represent individual ads. PoliticalAds2018.csv contains 659 entries and PoliticalAds2019.csv contains 3609 entries. Each dataset contains various columns including StartDate, EndDate, Spend, and CandidateBallotInformation, which contain relevant information on the run of the ad, impressions, spend, the organization that created the ad, the organization that paid, and targeting criteria.

The datasets contain both quantitative and categorical data. Columns such as Spend and Impressions were quantitative and stored as integers while columns like Organization Name, Gender, and Interests were nominal and stored as strings.

We noticed that some advertisements were associated with prominent campaigns but did not state this relationship in the CandidateBallotInformation column. We wanted to explore whether the missing values in CandidateBallotInfomation were random occurrences or a direct attempt to hinder transparency? Additionally, Snap Inc. details that an advertiser might leave the campaign end date empty if the ad runs indefinitely so we wanted to ask another question: what are the defining characteristics of ads that don't have a specified end date?

### 1.2 Cleaning and EDA

In order to begin our analysis and answer our question, we first had to clean our data. The first step we took was to concatenate the 2018 CSV and 2019 CSV into a singular dataframe which we called ads. We then converted the Start Date and End Date columns into datetime and made sure that all the datetimes were in PST (west coast, best coast). The timezone changes should have minimal effect on our analyses because at most they will be changing the date by one day. We then began to perform exploratory data analysis in order to understand the data and how we could use it to answer our question.

For univariate analysis, we looked at various columns to discern relevant statistics. For nominal columns like Gender and PayingAdvertiserName, we used value counts to see the distribution of counts and plotted the distribution. We wanted to see how ads target their audiences so we examined targeting criteria columns like Regions (Included)/(Excluded), Interests, AdvancedDemographics, etc which we identified from the data dictionary but realized that most ad campaigns leave these blank. We found that Electoral Districts (Excluded) had all NaN values meaning that no ads specified Electoral Districts to exclude. On the other hand, Regions (Included) and Language had the least amount of NaN with about 72% NaN for each.

We then chose to look at columns that had fewer NaN values. We looked at Currency Code and found that there are only 5 currencies; USD, EUR, GBP, CAD and AUD. Of these, USD is by far the most prevalent with 2,532 ads using USD.

We also looked at OrganizationName which had 365 unique values. We found that UnRestrict Minnesota was the most common and by visiting the url listed in CreativeProperties, we found that the ad was for a pro-choice organization. Media marketing companies such as Blueprint Interactive and ACRONYM and prominent US presidential campaigns such as Warren for President and Pete for America were in the top 25 most common OrganizationNames. Also, international organizations had many ads, such as Arbeiderpartiet i Bergen, which might mean that certain nations were having elections or other political and advocacy movements between 2018 and 2019.

CandidateBallotInformation also had many NaN values so we dropped those and then plotted its value counts. This helped us better understand which candidates and ballot initiatives an ad was associated with. We saw that General Election was the most common value with Warren for President following closely behind.

We looked at PayingAdvertiserName to understand the organizations paying for the political ads. We wanted to know what type of organization was likely to be the paying advertiser and what was their relationship with the political or advocacy issue/campaign? UnRestrict Minnesota was the most frequent value which matched the most frequent OrganizationName. 7 of the values matched with values in the top 25 OrganizationNames. This suggests that ads are often paid for by the organization they are associated with. We noticed that liberteeshop was listed in the top 25 PayingAdvertiserNames but liberteeshop.com was listed in the top 25 OrganizationNames indicating that slight differences in the name prevent us from matching even more values. However, not all the values match indicating that some ads may be paid for by outside organizations not directly connected to the campaign. We noted that Warren for President appeared in the top 25 values for OrganizationName, CandidateBallotInformation, and PayingAdvertiserName and we would want to explore data associated with Warren's campaign later on.

We also looked at the distribution of Gender among the ads. The data dictionary specifies that NaN is used to target all genders so we considered it a valid category. We also looked at the relative frequency of each category. Most ads are not targeting a specific gender but of the ones that do, there are almost three times as many ads targeting females than ads targeting males.

We also looked at quantitative columns. For these columns, we plotted the data, calculated the min, max, and mean values and identified the ads associated with these values. We first did this for the Spend column. We decided to limit the data to ads where the Currency Code was USD since that had the most values. Our plot had a steep curve which reveals that the majority of Snapchat ads cost much less than the maximum. In other words, the distribution skews toward the left. We saw that about 2,000 ads cost at most $1,000. We found the minimum of the spend column to be 0 and the maximum to be 451,244. The mean was about 1,974.96. The ad with the highest spend value is the General Mills ad and the ad with the lowest spend was associated with OpenPoll Inc. We noted that although there were 82 Impressions, the spend was 0 USD. Additionally, we noticed that there was a missing EndDate.

We followed the same steps for the Impressions column and saw that the distribution of values for the Impression column looks very similar to the distribution of the Spend column which further demonstrates our theory that the number of impressions correlates to the spend amount. However, the curve for Impressions is much steeper since many ads get millions of views. The minimum was 1, the maximum was 234,901,755, and the mean was about 702295.62. The minimum value came from an UnRestrict Minnesota ad with only 1 view. We noticed that this ad was also missing an end date. We were curious as to whether ad campaigns with less reach were more likely to leave out the EndDate.

We then performed bivariate analysis to identify possible associations between the columns. We were interested in understanding trends in targeting criteria and characteristics of ads missing CandidateBallotInformation.

To understand trends in targeting criteria, we chose to look at trends over time. We grouped StartDate by month and looked at the total campaigns by month. We also looked at the counts of campaigns targeting a single gender. We also looked at the proportion of ads that targeted a specific gender by month and we found that for all the months, female-targeted ads were the majority. We also looked at the total spending and impression on ads by Start Month. We noticed that the total spending and impressions by month have similar distributions and have a spike on November 2019. This correlates with the 2019 United States elections.

We then considered the characteristics of ads that were missing CandidateBallotInformation. Did these ads spend more or less than ads with CandidateBallotInformation? Did they have more Impressions? We saw that ads missing CandidateBallotInformation had higher Spend and Impression values in total.

We also wanted to know how ads with missing CandidateBallotInformation used targeting criteria. We looked at the proportion and counts of ads that used Interests and Regions (Included). It seemed that ads missing CandidateBallotInformation were less likely to include Interests but more likely to include Regions (Included).

We also looked at the distribution of genders for ads missing and not missing CandidateBallotInformation. We noticed that ads missing CandidateBallotInformation had a more varied distribution of genders and all ads targeting only males were missing CandidateBallotInformation.

Similarly, we wanted to look at the characteristics of ads without an EndDate. We first looked at their spending and impressions compared to ads with a defined EndDate. We found that on average, ads without a defined EndDate had much lower Spend and Impressions. This implies that ads without an EndDate are likely also ads with a smaller reach.

We followed the same steps to consider the usage of targeting criteria by ads with and without a defined EndDate. We looked at the proportion and counts of ads that used Interests and Regions (Included). It seemed that ads missing EndDate were almost equally likely to include Interests as ads not missing EndDate but almost twice as likely to include Regions (Included).

We also looked at the distribution of genders for ads with and without EndDate. We noticed that ads without EndDate were more likely to target specific genders.

We wanted to focus our investigation on the characteristics of ads missing CandidateBallotInformation or EndDate so we tried to uncover interesting aggregates regarding those columns. We first looked at the proportion of ads missing CandidateBallotInformation or EndDate by month.

We also looked at the occurrence of CountryCode according to CandidateBallotInformation and EndDate to see if international ads were more likely to be missing these columns. We only considered countries with 10 or more ads. We saw that many countries namely, Ireland, Kuwaid, Belgium, Denmark, Finland, Poland, India, and the Netherlands, never included CandidateBallot-Information. However, many countries followed close behind and ads from the US only included CandidateBallotInformation about 86% of the time. Additionally, the Netherlands was the most likely to not have an EndDate.

## 1.3 Assessment of Missingness

We first identified columns with missing data according to the data dictionary. We found that End-Date, CandidateBallotInformation, and CreativeProperties were the columns with missing data. To clarify, we concluded that CandidateBallotInformation had missing data because the data dictionary didn't specifically state that missing values represented the absence of a candidate promoted by the ad. To further support this conclusion, we found ads where Warren for President was listed as the paying advertiser and CreativeProperties linked to Warren's campaign website but Warren wasn't listed under CandidateBallotInformation.

We believe EndDate is NMAR because campaigns without a specified end date are less likely to report it and missing values don't have a specified meaning in the data dictionary. To make this data MAR, we would like to collect data on whether the campaign is intended to run indefinitely. To further support this assumption, we are using our hypothesis tests later in the project to determine if there are any defining characteristics of campaigns without an end date. We believe CandidateBallotInformation is not NMAR because the missingness can be explained by the ad not supporting a particular candidate or ballot initiative.

We believe CreativeProperties is not NMAR because having a URL isn't dependent on the content of the URL. For our missingness assessment, we focused on CandidateBallotInformation and the type of missingness in it to help address our first identified question: is CandidateBallotInfomation being empty a random occurrence or is it a direct attempt to hinder transparency?

We started by looking at some basic summary statistics for each column when CandidateBallot-Information was missing compared to the whole dataset. We identified Currency Code and Spend as columns we wanted to test CandidateBallotInformation's dependency on. Through our initial analysis, we do not think that CandidateBallotInformation is MD because there was no way to predict when the values were missing using other columns of the dataset.

Using a permutation test, we compared the distributions of Null and Non-Null values for CandidateBallotInformation on Currency Code. Our null hypothesis is that the two distributions of Currency Code are approximately equal and the alternative hypothesis is that the two distributions are significantly different. Our test statistic was total variation distance. The p-value for the permutation test was 0.0 and with a significance level of 0.05, we can reject the null hypothesis. Therefore, the two distributions are different, meaning that the missingness of CandidateBallot-Information is MAR dependent on the distribution of Currency Code. Upon a further look into the results of the test, we found that when CandidateBallotInformation is missing, the proportion of USD currency codes is higher, thus causing the greater total variation distance. We can say that there is a relationship between missing CandidateBallotInformation and campaigns that pay in USD that would be worth further exploring with more rigorous testing.

Using a permutation test, we compared the distributions of Null and Non-Null values for Candidate-BallotInformation to the distribution of Null and Non-Null values of Radius Targeting (Included).

Our null hypothesis is that the two distributions are approximately equal and the alternative hypothesis is that the two distributions are significantly different. Our test statistic was total variation distance. The p-value for the permutation test was 0.36 and with a significance level of 0.05, we can't reject the null hypothesis. Therefore, the two distributions aren't different, meaning that the missingness of CandidateBallotInformation isn't MAR on the distribution of Null and Non-Null values of Radius Targeting (Included).

This assessment helped us answer our first question. We determined that CandidateBallotInformation is not missing at random and the missingness of CandidateBallotInformation is in fact dependent on Currency Code.

## 1.4 Hypothesis Test

We made hypothesis tests to help us answer our earlier question: what are the defining characteristics of ads that don't have a specified end date?

First hypothesis test:
We wanted to see if there was any difference between the average spend for campaigns where the end date is missing compared to the average spend for all campaigns. To limit variables in our test, we only looked at campaigns that were paid in USD.

The null hypothesis: for campaigns that are paid in USD, the average spend of campaigns where the end date is missing is approximately equal to the average spend for all campaigns.
The alternative hypothesis: for campaigns that are paid in USD, the average spend of campaigns where the end date is missing is lower than the average spend for all campaigns.

Our test statistic was the mean and our p-value was 0.0. With a standard significance level of 0.05, we can reject the null hypothesis. We didn't use the KS-Statistic because with the mean test statistic, we could already tell that the distributions were different from each other. By rejecting the null hypothesis, we came to the conclusion that the average spend of campaigns where the end date is missing is lower than the average spend for all campaigns.

Second hypothesis test:
We wanted to see if there was any difference between the distribution of specified gender-targeted campaigns where the campaign end date is missing compared to the distribution of specified gender-targeted campaigns for all campaigns. As specified by the data dictionary, we considered campaigns where the targeted gender is null to be a category of its own since it refers to campaigns that included all genders.

The null hypothesis: the total variation distance of genders where the campaign end date is missing is approximately equal to the total variation distance of genders for all campaigns.
The alternate hypothesis: the total variation distance of genders where the campaign end date is missing is greater compared to the total variation distance of genders for all campaigns.

We did a permutation test using the distribution of proportions for ads targeting each gender category. Our test statistic was total variation distance and our p value is 0.0. With a standard significance level of 0.05, we can reject the null hypothesis. By rejecting the null hypothesis, we came to the conclusion that the distribution of genders where the campaign end date is missing is more varied compared to the distribution of genders for all campaigns. When further examined, this gender distribution variance stems from the significantly higher proportion of campaigns that didn't target a specific gender (Null).

These hypothesis tests helped us answer our second question. Ads with missing end date values spend less on their campaign and have greater total variation distance in their gender-targeting distribution.

### 1.5 Conclusion:

Here are summaries of our findings and how they answer our posed questions. Something to keep in mind with our analysis is that this is a purely observational study on the political ads of Snapchat and not a randomized controlled trial and therefore we can't assume causality or draw definitive conclusions from our results. Possible sources of error and uncertainty include: human data entry error, bias from data skew due to the overrepresentation of ads from 2019, and only analyzing campaigns that paid in USD for analysis on spending data.

Q1: Are the missing values in CandidateBallotInfomation random occurrences or a direct attempt to hinder transparency?
This assessment helped us answer our first question. We determined that CandidateBallotInformation is not missing at random and the missingness of CandidateBallotInformation is in fact dependent on Currency Code and not on Radius Targeting (Included).

For further exploration, one could web scrape the CreativeProperties URLs to determine what issues and platforms the advertisers are trying to promote and examine for any differences in content between ads that specify CandidateBallotInformation and those that don't.

Q2: What are the defining characteristics of ads that don't have a specified end date?
Hypothesis tests helped us answer our second question. Ads with missing end date values spend less on their campaign and are generally smaller, Additionally, these ads have greater total variation distance in their gender-targeting distribution, specifically a higher proportion of campaigns that didn't target a specific gender.

To discover more defining characteristics of these ads, we could run more tests on the data, specifically determining missing dependence on columns like Interests, Regions, and Locations. Additionally, one could send out a survey to the campaigns with no end date asking why they didn't fill out that value.

## 2 Code

### 2.1 Cleaning and EDA

[4]:
```
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
```

We first read in the Political Ads datasets from 2018 and 2019 then concatenated them to form a single dataset with all Snapchat ad campaigns in 2018 and 2019.

```
[5]: # reading in data sets
     ads_2018 = os.path.join('PoliticalAds2018.csv')
     ads18 = pd.read_csv(ads_2018)
     ads_2019 = os.path.join('PoliticalAds2019.csv')
     ads19 = pd.read_csv(ads_2019)

     # concat the data sets
     ads = pd.concat([ads18, ads19], ignore_index=True)
     ads.columns
```

```
[5]: Index(['ADID', 'CreativeUrl', 'Currency Code', 'Spend', 'Impressions',
            'StartDate', 'EndDate', 'OrganizationName', 'BillingAddress',
            'CandidateBallotInformation', 'PayingAdvertiserName', 'Gender',
            'AgeBracket', 'CountryCode', 'Regions (Included)', 'Regions (Excluded)',
            'Electoral Districts (Included)', 'Electoral Districts (Excluded)',
            'Radius Targeting (Included)', 'Radius Targeting (Excluded)',
            'Metros (Included)', 'Metros (Excluded)', 'Postal Codes (Included)',
            'Postal Codes (Excluded)', 'Location Categories (Included)',
            'Location Categories (Excluded)', 'Interests', 'OsType', 'Segments',
            'Language', 'AdvancedDemographics', 'Targeting Connection Type',
            'Targeting Carrier (ISP)', 'CreativeProperties'],
           dtype='object')
```

We then converted the StartDate and EndDate columns to datetime in PST.

```
[6]: # convert Start and End dates to date time objects + to PST
     ads['StartDate'] = pd.to_datetime(ads['StartDate']).dt.tz_convert(tz="America/
      ↪Los_Angeles")
     ads['EndDate'] = pd.to_datetime(ads['EndDate']).dt.tz_convert(tz="America/
      ↪Los_Angeles")
```

### 2.1.1 Univariate Analysis

We looked at the distributions and statistics for single columns. We considered different measures of center and spread to throughly understand the data.

For nominal columns, we looked at the distribution of value counts. First, we wanted to understand how these ads are able to reach their target group so we looked at columns such as Regions (Included)/(Excluded), Electoral Districts (Included)/(Excluded), Location Categories (Included)/(Excluded), Interests, AdvancedDemographics, etc. We quickly realized that most ad campaigns leave these blank so there are many NaN values. We created dictionaries for the counts and proportions of NaN values in these columns and saw that Region (Included) and language had the least NaN values with about 72% NaN for each.

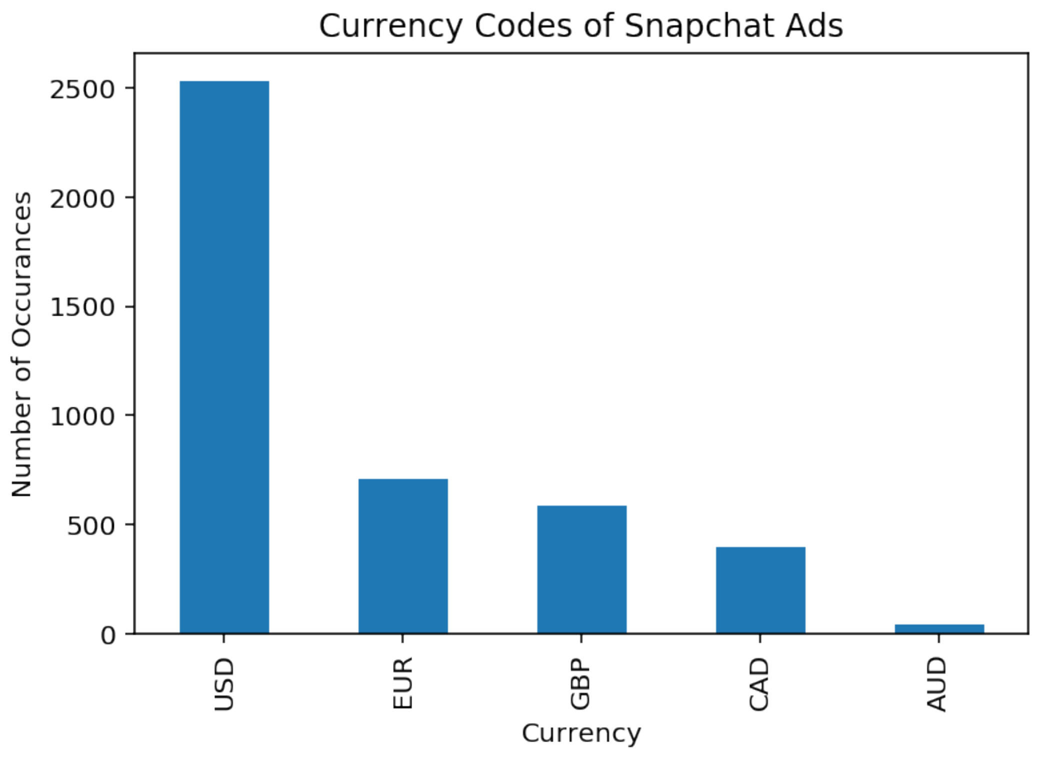
[7]:
```
#Creating a dictionary for the number of NaN values for each column of␣
↪targeting criteria
target_criteria = (['Regions (Included)', 'Regions (Excluded)', 'Electoral␣
↪Districts (Included)',
                    'Electoral Districts (Excluded)','Radius Targeting␣
↪(Included)', 'Radius Targeting (Excluded)',
                    'Metros (Included)', 'Metros (Excluded)','Postal Codes␣
↪(Included)','Postal Codes (Excluded)',
                    'Location Categories (Included)', 'Location Categories␣
↪(Excluded)', 'Interests', 'OsType', 'Segments',
                    'Language', 'AdvancedDemographics'])
na_counts = {}
for col in target_criteria:
    na_counts[col] = ads[col].isna().sum()

#dictionary for the proportion of NaN values in each column of targeting␣
↪criteria
na_prop = {key: na_counts[key]/ads.shape[0] for key in na_counts.keys()}
na_prop
```

[7]:
```
{'Regions (Included)': 0.7242268041237113,
 'Regions (Excluded)': 0.989456419868791,
 'Electoral Districts (Included)': 0.9847703842549204,
 'Electoral Districts (Excluded)': 1.0,
 'Radius Targeting (Included)': 0.9283036551077788,
 'Radius Targeting (Excluded)': 0.9971883786316776,
 'Metros (Included)': 0.9587628865979382,
 'Metros (Excluded)': 0.9960168697282099,
 'Postal Codes (Included)': 0.8024835988753515,
 'Postal Codes (Excluded)': 0.9725866916588566,
 'Location Categories (Included)': 0.9962511715089035,
 'Location Categories (Excluded)': 0.9995313964386129,
 'Interests': 0.7436738519212746,
 'OsType': 0.9939081537019682,
 'Segments': 0.2935801312089972,
 'Language': 0.729381443298969,
 'AdvancedDemographics': 0.9744611059044048}
```

We then chose to look at Currency Code, Organization Name, CandidateBallotInformation, PayingAdvertiserName, and Gender. We found that USD was the most common currency. We found it interesting that there were only 5 currencies though Snapchat is an international platform and came to the conclusion that Snap Inc likely only accepts payments in these currencies.

[8]:
```
#plotting the counts of Currency Code
barplot = (ads['Currency Code'].value_counts(dropna=False)
            .plot(kind='bar', title ='Currency Codes of Snapchat Ads'))
barplot.set_xlabel('Currency')
barplot.set_ylabel('Number of Occurances')
```

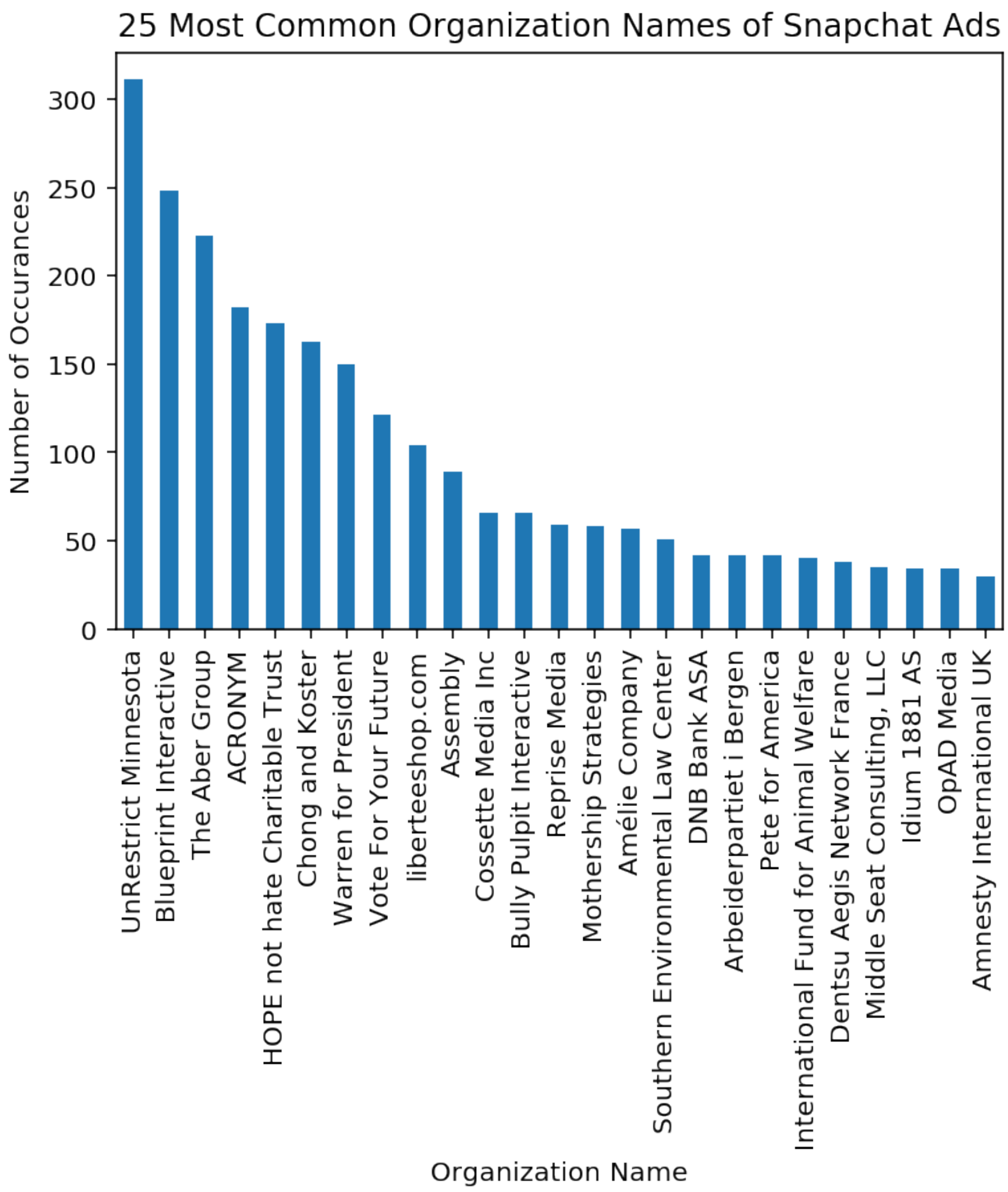
[8]: 
```
Text(0, 0.5, 'Number of Occurances')
```



We then looked at OrganizationName which had 365 unique values so for visualization purposes, we limited our barplot to the 25 most common values. We found that UnRestrict Minnesota was the most common and by visiting the url listed in CreativeProperties, we found that the ad was for a pro-choice organization. Media marketing companies such as Blueprint Interactive and ACRONYM were also listed. We also noticed prominent US presdential campaigns such as Warren for President and Pete for America were included in the top 25. Also, international organizations had many ads, such as Arbeiderpartiet i Bergen, which might mean that certain nations were having elections or other political and advocacy movements between 2018 and 2019.

[9]:
```
#number of unique Organization Names
len(ads['OrganizationName'].unique())  # = 25

#plotting the counts of OrginizationName for top 25 most frequent values
barplot = (ads['OrganizationName'].value_counts(dropna=False)[:25]
            .plot(kind='bar', title ='25 Most Common Organization Names of␣
 ↪Snapchat Ads'))
barplot.set_xlabel('Organization Name')
barplot.set_ylabel('Number of Occurances')
```

[9]:
```
Text(0, 0.5, 'Number of Occurances')
```



We also noticed that the most expensive ad in USD was also the most viewed ad indicating a dependent relatioship between Spend and Impressions (ie views). Upon inspection, we found it to be a General Mills ad with 234,901,755 impressions and cost $451,244.

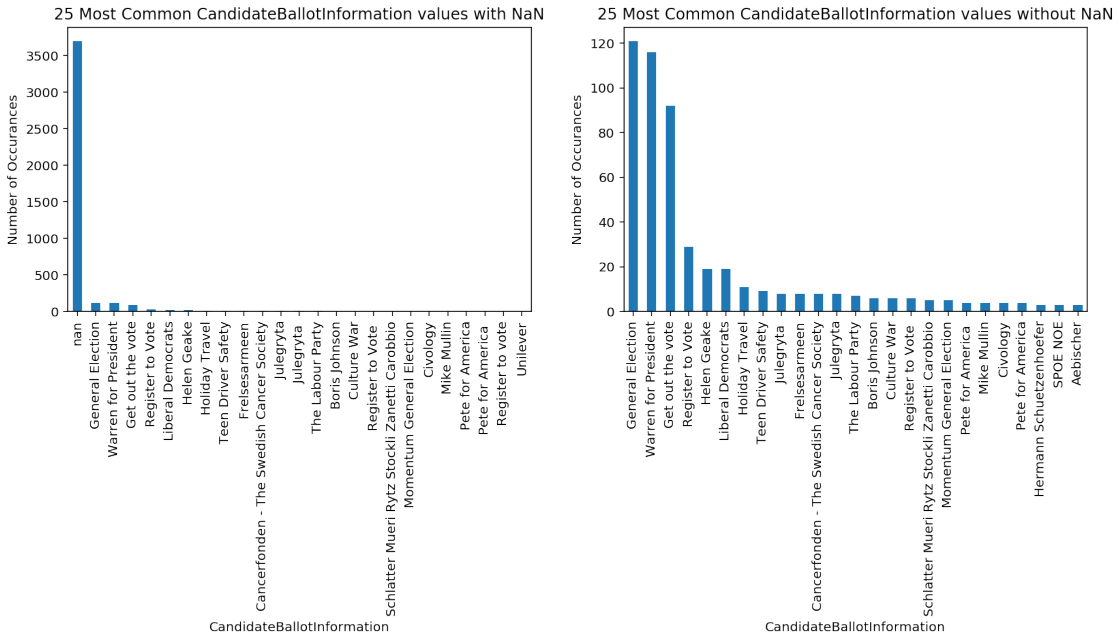
[10]:
```
ads[ads['Currency Code'] == 'USD']['Spend'].idxmax() == ads['Impressions'].
↪idxmax()
```

```
[10]: True
```

CandidateBallotInformation had 3698 NaN values so we plotted the value counts with and without NaN to better visualize which candidates and ballot initiatives an ad was associated with. Using the barplot without NaN values, we could see that General Election was the most common value with Warren for President following closely behind.

```
[11]: #number of unique CandidateBallotInformation
      len(ads['CandidateBallotInformation'].unique())  # = 72

      #number of NaN values for CandidateBallotInformation
      ads['CandidateBallotInformation'].isna().sum()  # = 3698

      fig, axes = plt.subplots(1, 2, figsize=(14, 4))

      #plotting the counts of CandidateBallotInformation for top 25 most frequent␣
       ↪values without NaN
      barplot = (ads['CandidateBallotInformation'].value_counts()[:25]
                 .plot(kind='bar', ax = axes[1], title ='25 Most Common␣
       ↪CandidateBallotInformation values without NaN'))
      barplot.set_xlabel('CandidateBallotInformation')
      barplot.set_ylabel('Number of Occurances')

      #plotting the counts of CandidateBallotInformation for top 25 most frequent␣
       ↪values including NaN
      barplot = (ads['CandidateBallotInformation'].value_counts(dropna=False)[:25]
                 .plot(kind='bar', ax = axes[0], title ='25 Most Common␣
       ↪CandidateBallotInformation values with NaN'))
      barplot.set_xlabel('CandidateBallotInformation')
      barplot.set_ylabel('Number of Occurances')
```

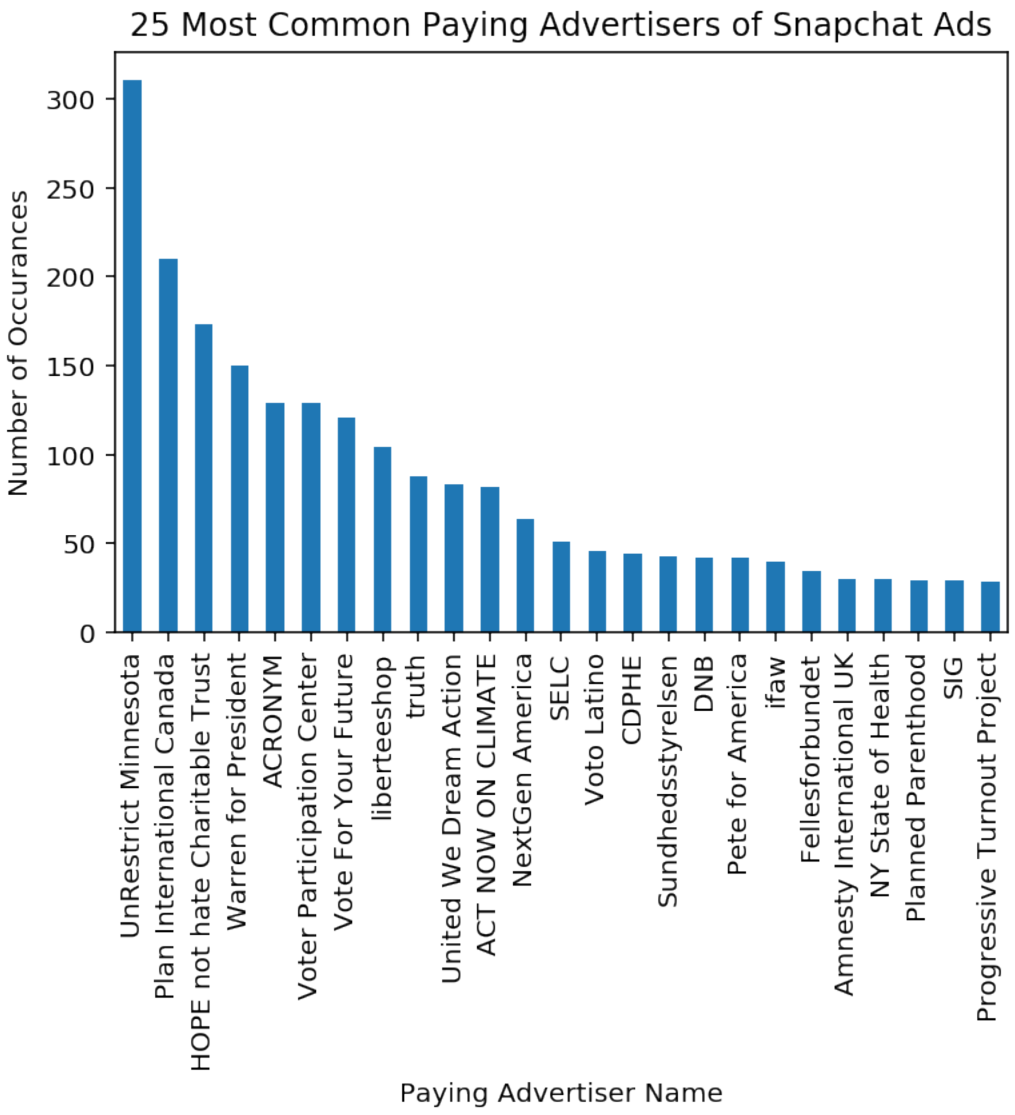
```
[11]: Text(0, 0.5, 'Number of Occurances')
```



We looked at PayingAdvertiserName to understand the organizations paying for the political ads. We wanted to know what type of organization was likely to be the paying advertiser and what was their relationship with the politcal or advocacy issue/campaign? There were 446 unique values so once again we limited our barplot to the top 25 values. UnRestrict Minnesota was the most frequent value which matched the most frequent OrganizationName. 7 of the values matched with values in the top 25 OrganizationNames. This suggests that ads are often paid for by the organization they are associated with. We noticed that liberteeshop was listed in the top 25 PayingAdvertiserNames but liberteeshop.com was listed in the top 25 OrganizationNames indicating that slight differences in the name prevent us from matching even more values. However, not all the values match indicating that some ads may be paid for by outside organizations not directly connected to the campaign. We noted that Warren for President appeared in the top 25 values for OrganizationName, CandidateBallotInformation, and PayingAdvertiserName and we would want to explore data associated with Warren's campaign later on.

```
[12]: #number of unique PayingAdvertiserName
len(ads['PayingAdvertiserName'].unique())  # = 446

#plotting the counts of PayingAdvertiserName for top 25 most frequent values
barplot = (ads['PayingAdvertiserName'].value_counts(dropna=False)[:25]
            .plot(kind='bar', title ='25 Most Common Paying Advertisers of␣
 ↪Snapchat Ads'))
barplot.set_xlabel('Paying Advertiser Name')
barplot.set_ylabel('Number of Occurances')

#count the number of matches between the top 25 OrganizationNames and top 25␣
 ↪PayingAdvertiserNames
```

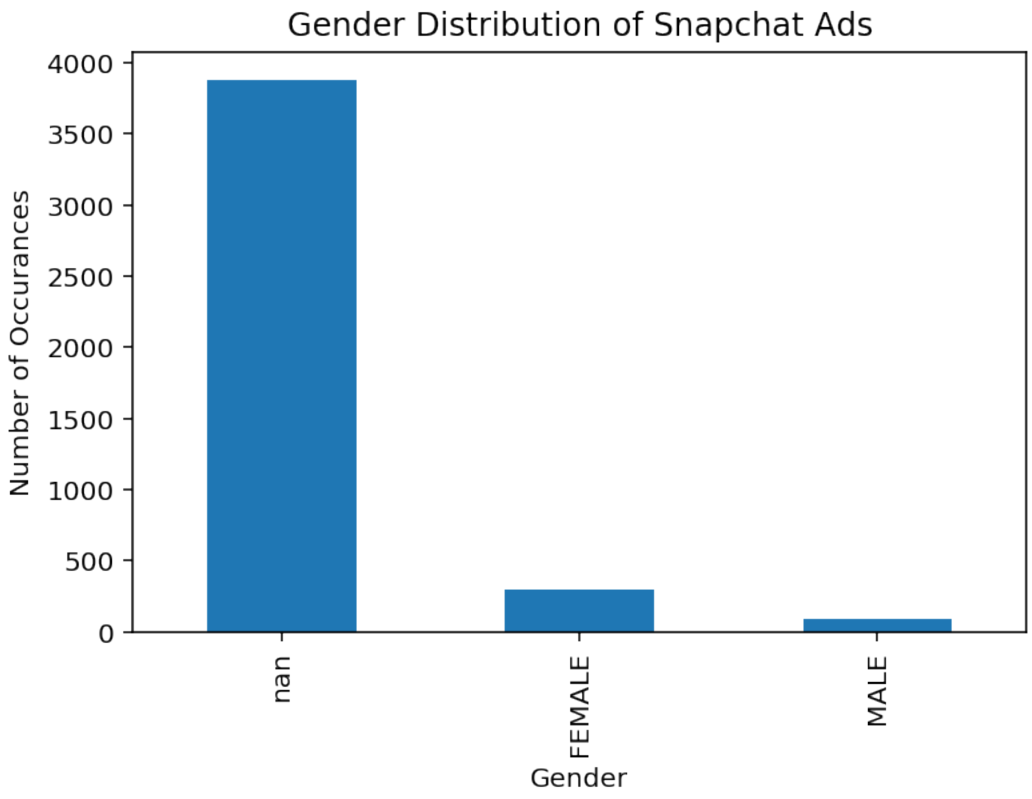
```python
matches = 0
for name in ads['PayingAdvertiserName'].value_counts(dropna=False)[:25].index:
    if name in ads['OrganizationName'].value_counts(dropna=False)[:25].index:
        matches += 1
# matches = 7
```



In the data dictionary, the Gender column is specified to have 3 valid values; MALE, FEMALE, and NaN where NaN is used to target all genders. Thus, we had to include NaN values when plotting the distribution. We also looked at the relative frequency of each category. Most ads are not targeting a specific gender but of the ones that do, there almost three times as many ads targeting females than ads targeting males.

```
[13]: #plotting the counts of Gender
      barplot = (ads['Gender'].value_counts(dropna=False)
                 .plot(kind='bar', title ='Gender Distribution of Snapchat Ads'))
      barplot.set_xlabel('Gender')
      barplot.set_ylabel('Number of Occurances')

      #dictionary of frequencies of each category
      prop = ({gen: ads['Gender'].value_counts()[gen]/ads.shape[0] for gen in␣
       ↪ads['Gender'].value_counts().index})
      prop['NaN'] = ads['Gender'].isna().mean()
      prop
```

```
[13]: {'FEMALE': 0.06865042174320525,
       'MALE': 0.02155576382380506,
       'NaN': 0.9097938144329897}
```

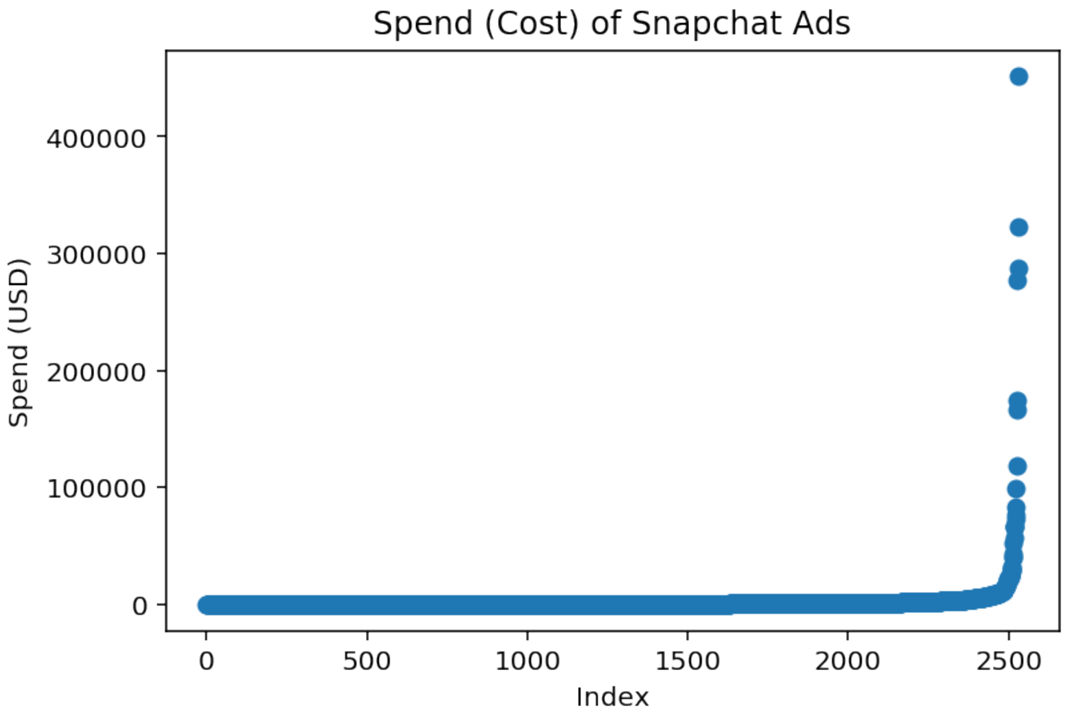


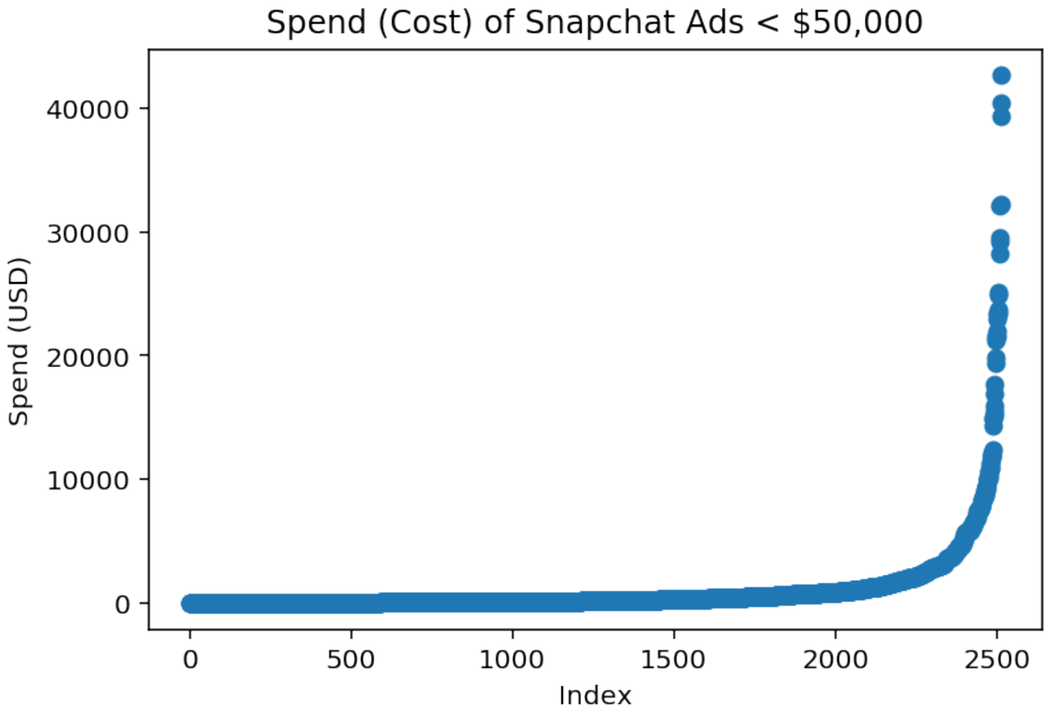
We also looked at quantitative columns. For these columns, we plotted the data, calculated the min, max, and mean values and identified the ads associated with these values. We first did this for the Spend column. Since the values in Spend are affected by Currency Code, we decided to limit the data to ads where the Currency Code was USD since that had the most values. To create our scatter plot, we sorted the dataset by the Spend column and reset the index. We then plotted the Spend columns and kept the index as the x axis. The first plot is the Spend of all ads in USD.

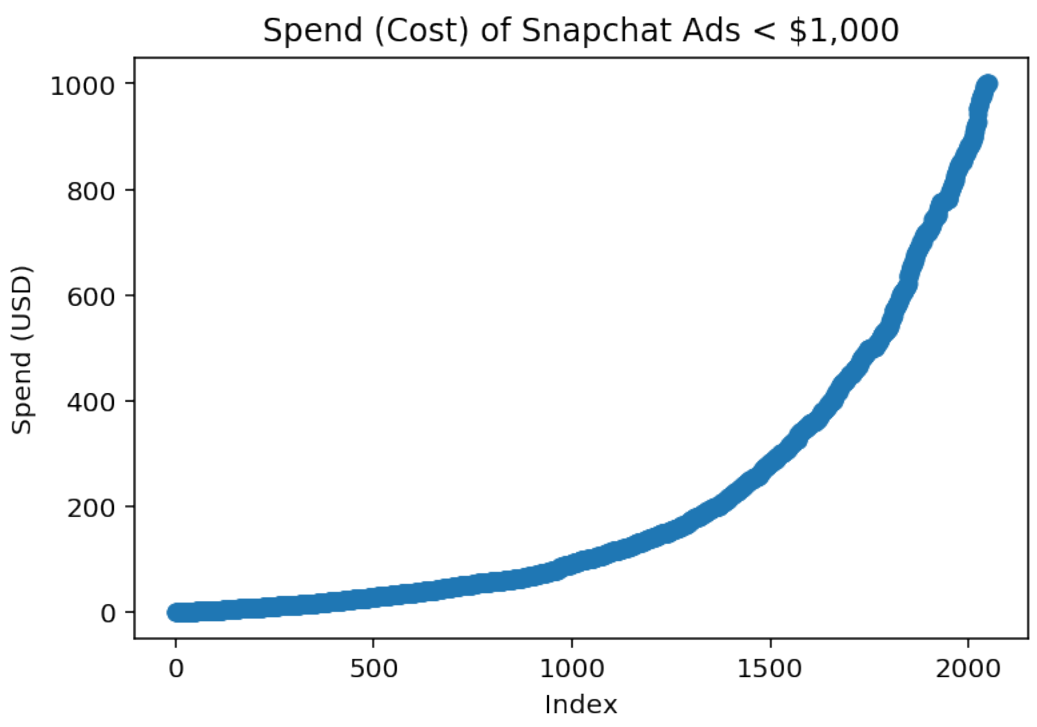
The steep curve reveals that the majority of Snapchat ads cost much less than the maximum. In other words, the distribution skews toward the left. To better understand the Spend values of less costly ads, we narrowed the range of the plot to ads that cost less than 50,000 USD and 1,000 USD. Looking at the third plot shows that about 2,000 ads cost at most $1,000

[14]:
```
#scatter plot of Spend column with x axis set to index
#selecting ads where Currency Code is USD
df = ads[ads['Currency Code'] == 'USD']
df = df.sort_values(by='Spend').reset_index()

plot1 = plt.figure(1)
plt.scatter(df.index, df['Spend'])
plt.title('Spend (Cost) of Snapchat Ads')
plt.xlabel('Index')
plt.ylabel('Spend (USD)')

#only looking at ads with a Spend under or at 50,000
df['Spend'] = df[df['Spend'] <= 50000]['Spend']

plot1 = plt.figure(2)
plt.scatter(df.index, df['Spend'])
plt.title('Spend (Cost) of Snapchat Ads < $50,000')
plt.xlabel('Index')
plt.ylabel('Spend (USD)')

#only looking at ads with a Spend under or at 1,000
df['Spend'] = df[df['Spend'] <= 1000]['Spend']

plot1 = plt.figure(3)
plt.scatter(df.index, df['Spend'])
plt.title('Spend (Cost) of Snapchat Ads < $1,000')
plt.xlabel('Index')
plt.ylabel('Spend (USD)')
```

[14]:
```
Text(0, 0.5, 'Spend (USD)')
```


Spend (Cost) of Snapchat Ads
Spend (USD)
400000
300000
200000
100000
0
0
500
1000
1500
2000
2500
Index


Spend (Cost) of Snapchat Ads < $50,000
Spend (USD)
40000
30000
20000
10000
0
0
500
1000
1500
2000
2500
Index



We then looked at the min, max, and mean values of the Spend column. Once again, we limited out data to ads in USD. These statistics show the large range of Spend. The mean is about 1,974 USD meaning that 50% of the data is less than that value. This number is probably this high due to the extremely high max and other extremely large values in the dataset that skew the mean. The ad with the highest spend value is the General Mills ad and the ad with the lowest spend was associated with OpenPoll Inc. We noted that although there were 82 Impressions, the spend was 0 USD. Additionally, we noticed that there was a missing EndDate.

```
[15]: #restrict data to ads with a Currency Code of USD
df = ads[ads['Currency Code'] == 'USD']
df = df.reset_index(drop=True)

#compile min, max, and mean
stats = {'min':df['Spend'].min(), 'max': df['Spend'].max(), 'mean':df['Spend'].
 ↪mean()}
print(stats)

#find the ads associated with the min and max
df = df.iloc[[df['Spend'].idxmin(), df['Spend'].idxmax()]][['OrganizationName',␣
 ↪'Spend', 'StartDate', 'EndDate', 'PayingAdvertiserName', 'Impressions']]
df.index = ['min', 'max']
df
```

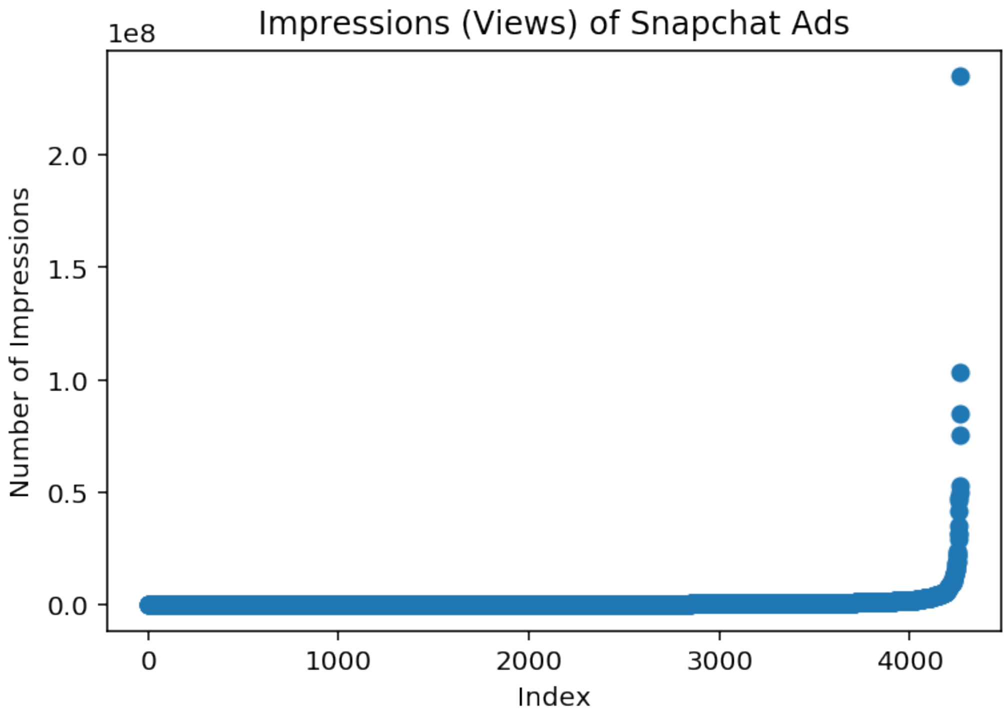
```
{'min': 0, 'max': 451244, 'mean': 1974.9640600315956}
```

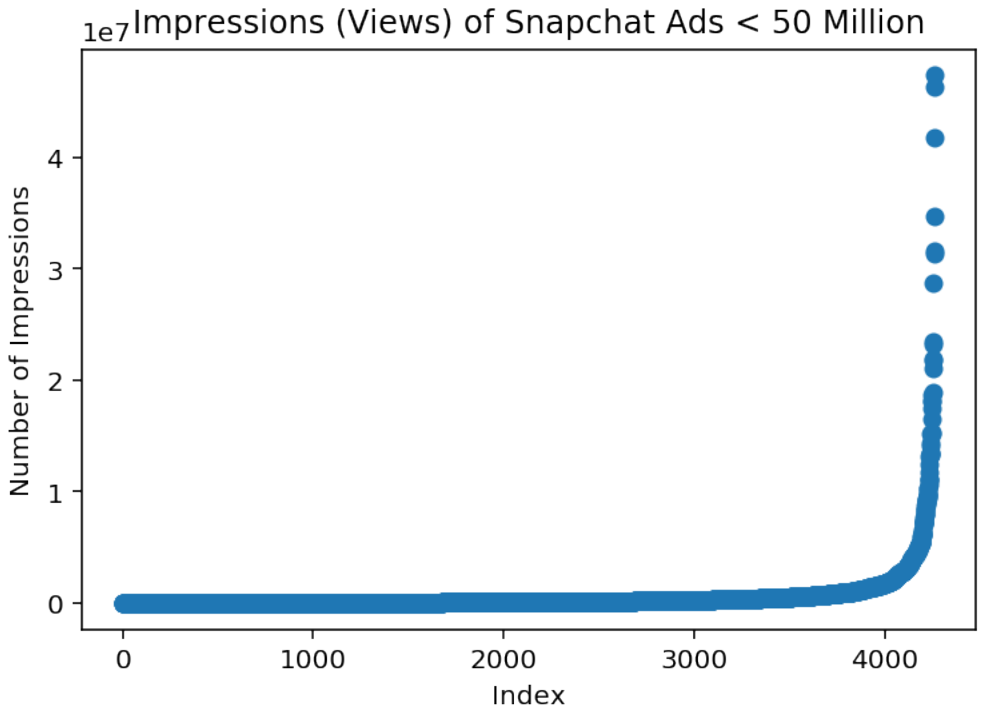
```
[15]:     OrganizationName   Spend                 StartDate  \
     min      OpenPoll Inc       0 2018-10-08 18:34:31-07:00
     max     General Mills  451244 2019-10-01 08:00:00-07:00

                          EndDate PayingAdvertiserName  Impressions
     min                      NaT    Students for Trump           82
     max 2019-12-03 21:59:00-08:00        general mills    234901755
```

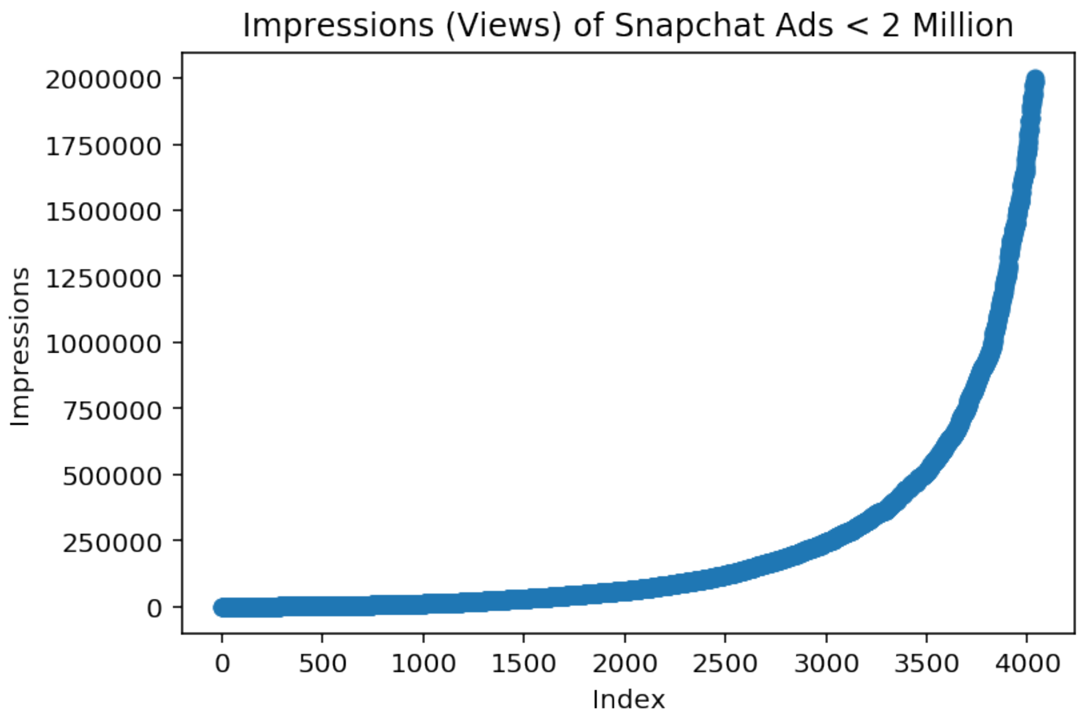
The distribution of values for the Impression column looks very similar to the distribution of the Spend column which further demonstrates our theory that the number of impressions correlates to the spend amount. However, the curve for Impressions is much steeper since many ads get millions of views.

```
[16]: #scatter plot of Impressions column with x axis set to index
df = ads.sort_values(by='Impressions').reset_index()

plot1 = plt.figure(1)
plt.scatter(df.index, df['Impressions'])
plt.title('Impressions (Views) of Snapchat Ads')
plt.xlabel('Index')
plt.ylabel('Number of Impressions')

#only looking at ads with Impressions under or at 50 Million
df['Impressions'] = df[df['Impressions'] <= 50000000]['Impressions']

plot1 = plt.figure(2)
plt.scatter(df.index, df['Impressions'])
plt.title('Impressions (Views) of Snapchat Ads < 50 Million')
plt.xlabel('Index')
plt.ylabel('Number of Impressions')

#only looking at ads with Impressions under or at 2 Million
df['Impressions'] = df[df['Impressions'] <= 2000000]['Impressions']

plot1 = plt.figure(3)
plt.scatter(df.index, df['Impressions'])
plt.title('Impressions (Views) of Snapchat Ads < 2 Million')
plt.xlabel('Index')
plt.ylabel('Impressions')
```

```
[16]: Text(0, 0.5, 'Impressions')
```


Impressions (Views) of Snapchat Ads
1e8
Number of Impressions
0.0
0.5
1.0
1.5
2.0
0
1000
2000
3000
4000
Index


Impressions (Views) of Snapchat Ads < 50 Million
1e7
Number of Impressions
0
1
2
3
4
0
1000
2000
3000
4000
Index



We found the min, max, and mean values of Impressions. The mean is much lower than than the maximum meaning that there are many ads fall below the mean which is clearly evident from the scatter plots above. We also found the ads associated with the min and the max. The maxaimum value of Impressions once again came the General Mills ad which had 234,901,755 views. The minimum value came from an UnRestrict Minnesota ad with only 1 view. We noticed that this ad was also missing an end date. We were curious as to whether ad campaigns with less reach were more likely to leave out the EndDate.

[17]:
```
df = ads.copy()

#compile min, max, and mean
stats = {'min':df['Impressions'].min(), 'max': df['Impressions'].max(), 'mean':
 ↪df['Impressions'].mean()}
print(stats)

#find the ads associated with the min and max
df = df.iloc[[df['Impressions'].idxmin(), df['Impressions'].
 ↪idxmax()]][['OrganizationName', 'Spend', 'StartDate', 'EndDate',␣
 ↪'PayingAdvertiserName', 'Impressions']]
df.index = ['min', 'max']
df
```

```
{'min': 1, 'max': 234901755, 'mean': 702295.6237113402}
```

```
[17]:          OrganizationName   Spend                 StartDate  \
      min  UnRestrict Minnesota       0 2019-09-17 14:21:41-07:00
      max         General Mills  451244 2019-10-01 08:00:00-07:00

                            EndDate  PayingAdvertiserName  Impressions
      min                       NaT  UnRestrict Minnesota            1
      max 2019-12-03 21:59:00-08:00         general mills    234901755
```

### 2.1.2 Bivariate Analysis

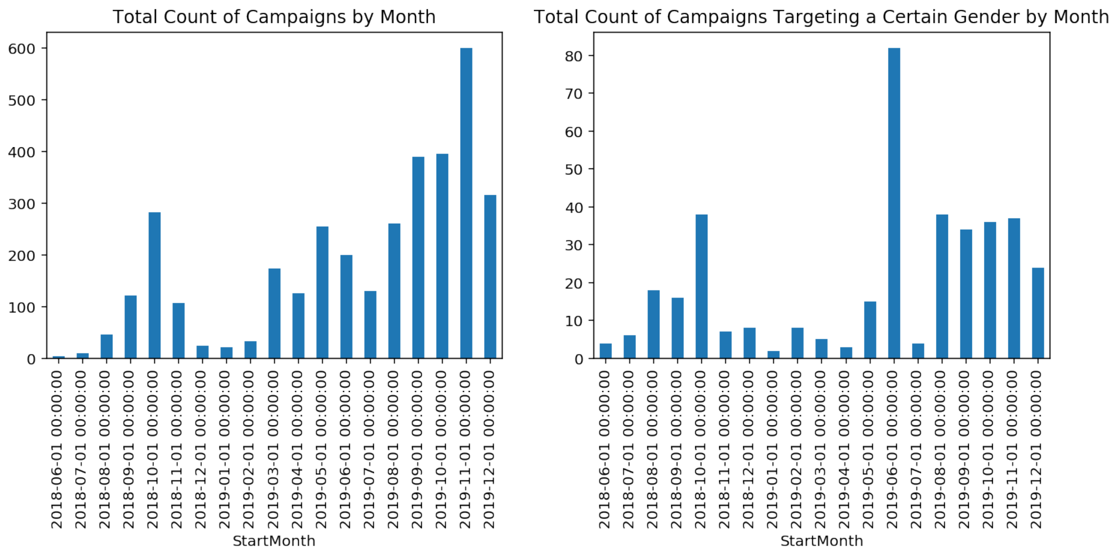
We then performed bivariate analysis to identify possible associations between the columns. We were interested in understanding trends in targeting criteria and characteristics of ads are missing CandidateBallotInformation. To understand trends in targeting criteria, we chose to look at trends over time. We grouped StartDate by month and looked at the total campaigns by month. We also looked at the counts of campaigns targeting a single gender.

```
[18]: fig, axes = plt.subplots(1, 2, figsize=(12, 4))
      relevant = ads[['StartDate', 'EndDate', 'Spend', 'Currency Code', 'CountryCode',
                      'Regions (Included)', 'Regions (Excluded)', 'Language',
                      'Gender', 'AgeBracket', 'Interests', 'AdvancedDemographics',␣
       ↪'Impressions', 'CandidateBallotInformation']]

      # total campaigns by start month
      #adding a start month column
      relevant['StartMonth'] = relevant['StartDate'].dt.month.astype(str) + "/"␣
       ↪+relevant['StartDate'].dt.year.astype(str)
      relevant['StartMonth'] = pd.to_datetime(relevant['StartMonth'])
      campaigns_over_time = relevant.groupby(['StartMonth'])['EndDate'].count()
      campaigns_over_time.plot(kind='bar', ax = axes[0], title='Total Count of␣
       ↪Campaigns by Month')

      #campaigns targeting a gender
      gender_over_time = relevant.groupby(['StartMonth', 'Gender'])['Gender'].count()
      reset = gender_over_time.to_frame().rename(columns={'Gender': 'Count'}).
       ↪reset_index()
      by_month = reset.groupby('StartMonth')['Count'].sum().reset_index(drop=True)
      plot_all_gender = reset.groupby('StartMonth')['Count'].sum()
      plot_all_gender.plot(kind='bar', ax = axes[1], title = 'Total Count of␣
       ↪Campaigns Targeting a Certain Gender by Month')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82b3d0f630>
```

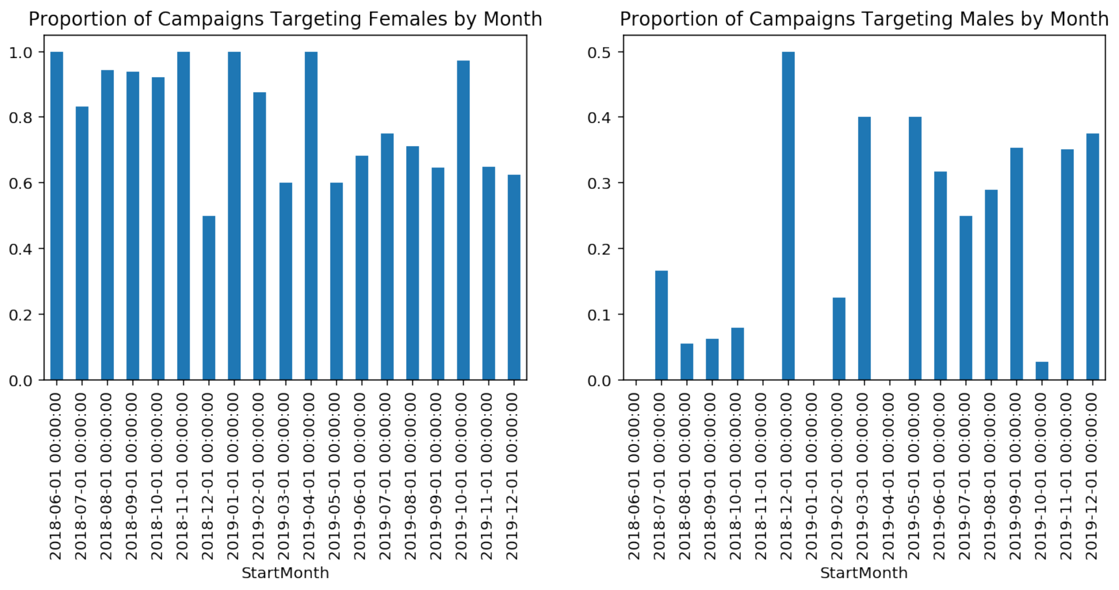


Out of the ads that target a specific gender, we then looked at the proportion of ads that target each gender by month.

[19]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

# female proportion graph
female = reset[reset["Gender"] == "FEMALE"].reset_index(drop=True)
ts = female['Count'].copy() / by_month
ts.index = female['StartMonth']
ts.plot(kind='bar', ax = axes[0], title = 'Proportion of Campaigns Targeting␣
 ↪Females by Month')

# male proportion graph
male = reset[reset["Gender"] == "MALE"]
by_month2 = reset.groupby('StartMonth')['Count'].sum().to_frame().reset_index()
```

```
ts2 = male.merge(by_month2, how='outer', on='StartMonth').fillna(0).
 ↪sort_values(by='StartMonth')['Count_x'].reset_index(drop=True)
ts2.index = female['StartMonth']
by_month_with_date = pd.Series(by_month.values, index= female['StartMonth'])
ts2 = ts2 / by_month_with_date
ts2.plot(kind='bar', ax=axes[1], title = 'Proportion of Campaigns Targeting␣
 ↪Males by Month')
```

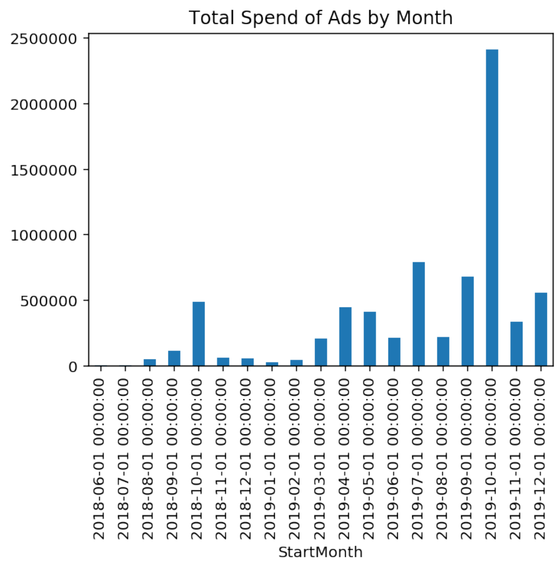
[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82af4434e0>
```

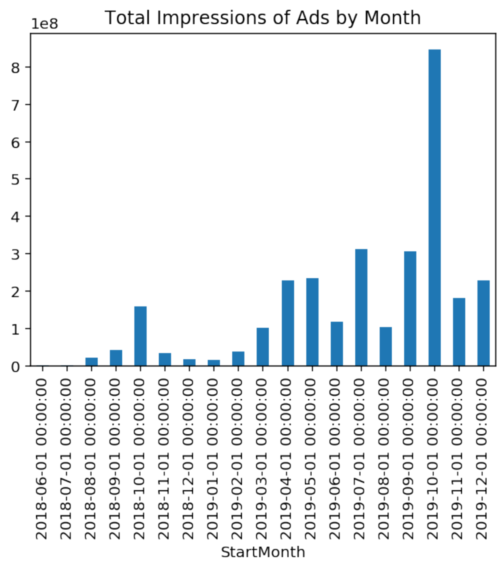


We also looked at the total spending and impression on ads by Start Month. We noticed that the total spending and impressions by month have similar distributions and have a spike on November 2019. This correlates with the 2019 United States elections.

[20]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

#plotting the total spending by month
spend_over_time = relevant.groupby(['StartMonth'])['Spend'].sum()
spend_over_time.plot(kind='bar', ax=axes[0], title = 'Total Spend of Ads by␣
 ↪Month')

#plotting the total impressions by month
impressions_over_time = relevant.groupby(['StartMonth'])['Impressions'].sum()
impressions_over_time.plot(kind='bar', ax=axes[1], title='Total Impressions of␣
 ↪Ads by Month')
```

[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82af2a5128>
```





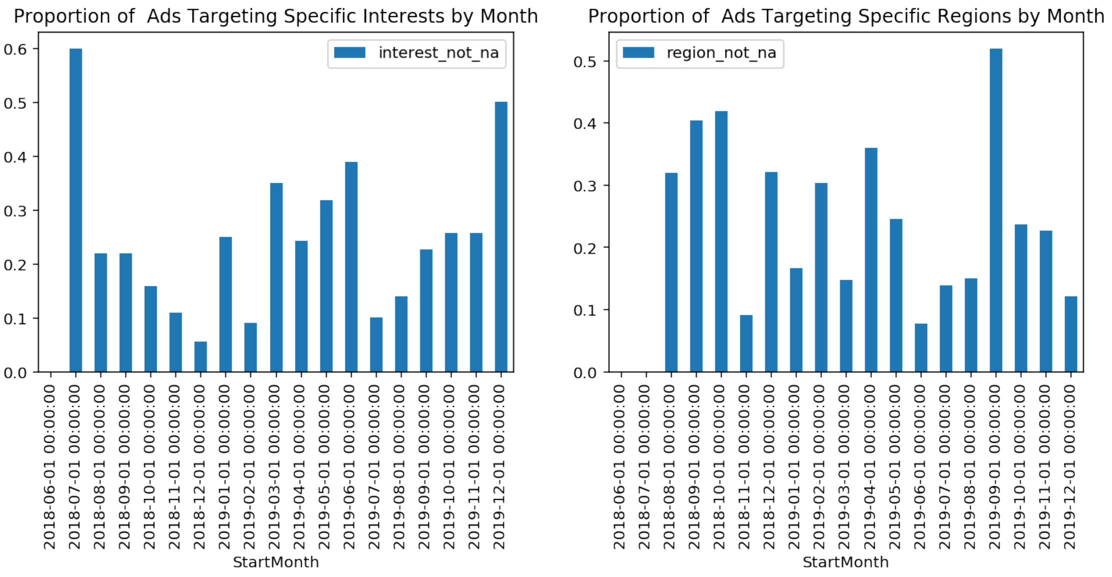
We wanted to see if ads, on average, were more general closer to major elections perhaps in order to target more people. In other words, we wanted to see when other targeting criteria such as Interests and Regions (Included) were used. It was difficult to come to a definite conclusion from these plots.

[21]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

#plotting number of ads with a non-null value for Interests by month
relevant['interest_not_na'] = relevant['Interests'].notna()
interest_over_times = relevant.groupby('StartMonth').agg({'interest_not_na':
 ↪'mean'})
interest_over_times.plot(kind='bar', ax=axes[0], title='Proportion of  Ads␣
 ↪Targeting Specific Interests by Month')

#plotting number of ads with a non-null value for Regions (Included) by month
relevant['region_not_na'] = relevant['Regions (Included)'].notna()
interest_over_times = relevant.groupby('StartMonth').agg({'region_not_na':
 ↪'mean'})
interest_over_times.plot(kind='bar', ax=axes[1], title='Proportion of  Ads␣
 ↪Targeting Specific Regions by Month')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
```

```
until
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82af1860b8>
```



We then considered the characteristics of ads that were missing CandidateBallotInformation. Did these ads spend more or less than ads with CandidateBallotInformation? Did they have more Impressions? We saw that ads missing CandidateBallotInformation had higher Spend and Impression values in total.

[22]:
```
#total and average spending and impressions of ads missing and not missing␣
↪CandidateBallotInformation
relevant['ballot_info_na'] = relevant['CandidateBallotInformation'].isna()
table = relevant.groupby('ballot_info_na').agg({'Spend':['count','sum',␣
↪'mean'], 'Impressions':['sum', 'mean']})
table
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[22]:
```
                     Spend                       Impressions
                       sum         mean count           sum           mean
ballot_info_na
False               588998  1033.329825   570     203656445  357292.008772
True               6555512  1772.718226  3698    2793741277  755473.574094
```

We also wanted to know how ads with missing CandidateBallotInformation used targeting criteria. We looked at the proportion and counts of ads that used Interests and Regions (Included). We found that about 23% of ads missing CandidateBallotInformation included Interests while 41% of ads not missing it would include Interests. For Regions (Included), we found that 29% of ads missing CandidateBallotInformation and 17% not missing CandidateBallotInformation would include that column. It seemed that ads missing CandidateBallotInformation were less likely to include Interests but more likely to include Regions (Included).

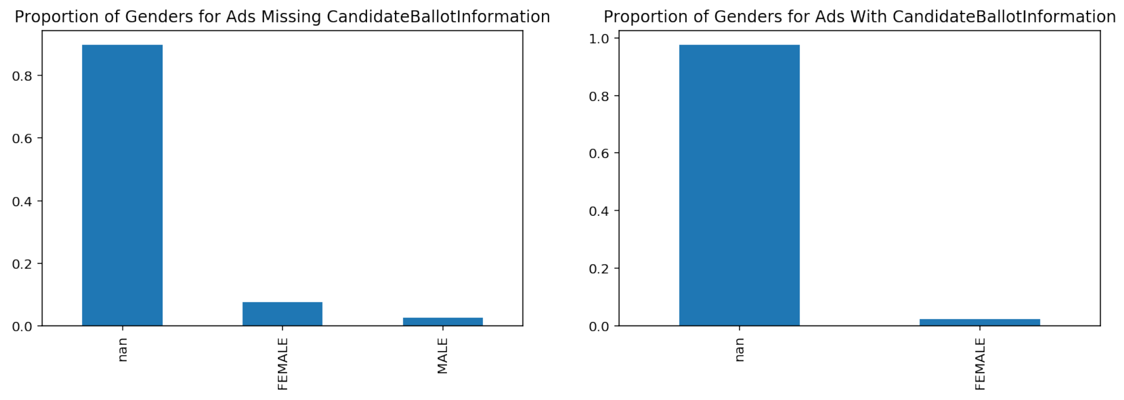
[23]:
```python
#proportion of ads that have non-null values for Interests and Region␣
↪(Included) by whether CandidateBallotInformation
#is missing
table = relevant.groupby('ballot_info_na').agg({'interest_not_na':['count',␣
↪'mean'], 'region_not_na':['mean']})
table
```

[23]:
```
               interest_not_na           region_not_na
                         count      mean         count      mean
ballot_info_na
False                      570  0.410526           570  0.168421
True                      3698  0.232558          3698  0.292320
```

We also looked at the distribution of genders for ads missing and not missing CandidateBallotInformation. We noticed that ads missing CandidateBallotInformation had a more varied distribution of genders and all ads targeting only males were missing CandidateBallotInformation.

[24]:
```python
fig, axes = plt.subplots(1, 2, figsize=(14, 4))
#plot of genders for ads without CandidateBallotInformation
((ads[ads['CandidateBallotInformation'].isna()]['Gender'].
↪value_counts(dropna=False)/ads['CandidateBallotInformation']
  .isna().sum()).plot(kind='bar', ax=axes[0], title='Proportion of Genders for␣
↪Ads Missing CandidateBallotInformation'))

#plot of genders for ads with CandidateBallotInformation
((ads[ads['CandidateBallotInformation'].notna()]['Gender'].
↪value_counts(dropna=False)/ads['CandidateBallotInformation']
  .notna().sum()).plot(kind='bar', ax=axes[1], title='Proportion of Genders for␣
↪Ads With CandidateBallotInformation'))
```

```
[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f82bbfb9b00>
```



Similarly, we wanted to look at the characteristics of ads without an EndDate. We first looked at their spending and impressions compared to ads with a defined EndDate. We found that on average, ads without a defined EndDate had much lower Spend and Impressions. This implies that ads without an EndDate are likely also ads with a smaller reach.

```
[25]: #total and average spending and impressions of ads missing and not missing␣
       ↪EndDate
      relevant['enddate_na'] = relevant['EndDate'].isna()
      table = relevant.groupby('enddate_na').agg({'Spend':['sum', 'mean', 'count'],␣
       ↪'Impressions':['sum', 'mean']})
      table
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
[25]:              Spend                        Impressions
                     sum         mean count          sum           mean
      enddate_na
      False      6738442  1930.232598  3491   2726562187  781026.120596
      True        406068   522.610039   777    270835535  348565.682111
```

We then looked at the usage of targeting criteria by ads with and without a defined EndDate. We looked at the proportion and counts of ads that used Interests and Regions (Included). We found that about 23% of ads missing EndDate included Interests while about 26% of ads not missing it would include Interests. For Regions (Included), we found that 47% of ads missing EndDate and 23% not missing EndDate would include that column. It seemed that ads missing EndDate were almost equally likely to include Interests as ads not missing EndDate but almost twice as likely to include Regions (Included).

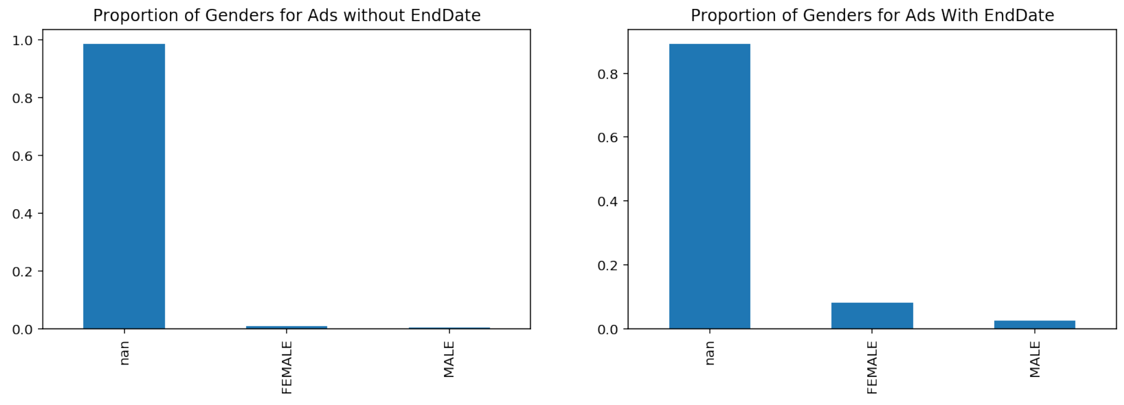
```
[26]: #proportion of ads that have non-null values for Interests and Region␣
       ↪(Included) by whether EndDate is missing
      table = relevant.groupby('enddate_na').agg({'interest_not_na':['count',␣
       ↪'mean'], 'region_not_na': 'mean'})
      table
```

```
[26]:            interest_not_na           region_not_na
                           count      mean          mean
      enddate_na
      False                 3491  0.260670      0.232885
      True                   777  0.236808      0.468468
```

We also looked at the distribution of genders for ads with and without EndDate. We noticed that ads without EndDate were more likely to target specific genders.

```
[27]: fig, axes = plt.subplots(1, 2, figsize=(14, 4))
      #plot of genders for ads without EndDate
      ((ads[ads['EndDate'].isna()]['Gender'].value_counts(dropna=False)/ads['EndDate']
        .isna().sum()).plot(kind='bar', ax=axes[0], title='Proportion of Genders for␣
       ↪Ads without EndDate'))

      #plot of genders for ads with defined EndDate
      ((ads[ads['EndDate'].notna()]['Gender'].value_counts(dropna=False)/
       ↪ads['EndDate']
        .notna().sum()).plot(kind='bar', ax=axes[1], title='Proportion of Genders for␣
       ↪Ads With EndDate'))
```

```
[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7f82b3fab1d0>
```

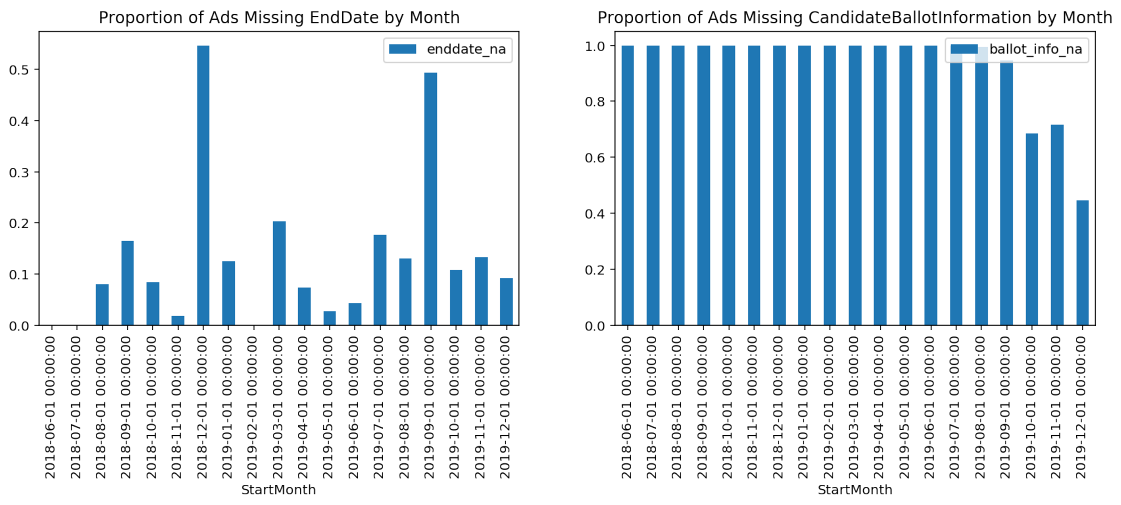


### 2.1.3 Interesting Aggregates

We wanted to focus our investigation on the characteristcs of ads missing CandidateBallotInformation or EndDate so we tried to uncover interesting aggregates regarding those columns. We first looked at the proportion of ads missing CandidateBallotInformation or EndDate by month.

[28]:
```
fig, axes = plt.subplots(1, 2, figsize=(14, 4))
(relevant.groupby('StartMonth').agg({'enddate_na':'mean'})
 .plot(kind='bar', ax=axes[0], title='Proportion of Ads Missing EndDate by␣
 ↪Month'))

(relevant.groupby('StartMonth').agg({'ballot_info_na':'mean'})
 .plot(kind='bar', ax=axes[1], title='Proportion of Ads Missing␣
 ↪CandidateBallotInformation by Month'))
```

[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82aefe7b00>
```



We also looked at the occurance of CountryCode according to CandidateBallotInformation and EndDate to see if international ads were more likely to be missing these columns. We only considered countries with 10 or more ads. We saw that many countries namely, Ireland, Kuwaid, Belgium, Denmark, Finland, Poland, India, and the Netherlands, never included CandidateBallotInformation. However, many countries followed close behind and ads from the US only included CandidateBallotInformation about 86% of the time. Additionally, Netherlands was the most likely to not have an EndDate.

[29]:
```
#counts of ads by country that are missing/not missing␣
 ↪CandidateBallotInformation
country_counts = (relevant.groupby(['ballot_info_na','CountryCode']).
 ↪agg({'CountryCode':'count'})
                           .rename(columns={'CountryCode':'Count'}).reset_index())
```

```
table = country_counts.pivot(index='CountryCode', columns='ballot_info_na',␣
↪values='Count').fillna(0)
#adding a column for proportion of ads missing CandidateBallotInformation by␣
↪country
table['proportion'] = table[True]/(table[False]+table[True])
#selecting countries with at least 10 ads and sorting by proportion column
table[table[False]+table[True] > 10].sort_values('proportion', ascending=False)
```

[29]:
```
ballot_info_na  False    True  proportion
CountryCode
ireland           0.0    26.0    1.000000
kuwait            0.0    35.0    1.000000
belgium           0.0    28.0    1.000000
denmark           0.0   105.0    1.000000
finland           0.0    35.0    1.000000
poland            0.0    14.0    1.000000
india             0.0    18.0    1.000000
netherlands       0.0    79.0    1.000000
canada            1.0   398.0    0.997494
france            1.0    71.0    0.986111
australia         1.0    35.0    0.972222
norway           24.0   425.0    0.946548
germany           1.0    12.0    0.923077
united states   312.0  1975.0    0.863577
austria           8.0    29.0    0.783784
sweden            9.0    28.0    0.756757
united kingdom  195.0   327.0    0.626437
switzerland      17.0    14.0    0.451613
```

[30]:
```
#counts of ads by country that are missing/not missing EndDate
country_counts = (relevant.groupby(['enddate_na','CountryCode']).
↪agg({'CountryCode':'count'})
                  .rename(columns={'CountryCode':'Count'}).reset_index())
table = country_counts.pivot(index='CountryCode', columns='enddate_na',␣
↪values='Count').fillna(0)
#adding a column for proportion of ads missing CandidateBallotInformation by␣
↪country
table['proportion'] = table[True]/(table[False]+table[True])
#selecting countries with at least 10 ads and sorting by proportion column
table[table[False]+table[True] > 10].sort_values('proportion', ascending=False)
```

[30]:
```
enddate_na       False   True  proportion
CountryCode
netherlands       39.0   40.0    0.506329
kuwait            23.0   12.0    0.342857
australia         26.0   10.0    0.277778
united states   1664.0  623.0    0.272409
```

```
belgium          25.0   3.0   0.107143
canada          373.0  26.0   0.065163
norway          423.0  26.0   0.057906
united kingdom  494.0  28.0   0.053640
france           69.0   3.0   0.041667
switzerland      30.0   1.0   0.032258
finland          34.0   1.0   0.028571
denmark         104.0   1.0   0.009524
germany          13.0   0.0   0.000000
india            18.0   0.0   0.000000
austria          37.0   0.0   0.000000
poland           14.0   0.0   0.000000
sweden           37.0   0.0   0.000000
ireland          26.0   0.0   0.000000
```

## 2.2 Assessment of Missingness

**We started by doing a cursory analysis of the dependence of columns with missingness on other columns** This helped us identify which columns were and weren't NMAR (Warning - Long Output)

[31]:
```python
# specify the column to be analyzed
missing_column = 'CandidateBallotInformation'
no_end = ads[ads[missing_column].isna()]

# for each column, calculate the following stats for the whole column and just␣
↪the data in the column when the
# analyzed column is NaN
for col in ads.columns:

    # if the column has quantitative data, calculate the proportion missing,␣
↪mean, median, max, and min
    if (ads[col].dtype == np.int64) | (ads[col].dtype == np.float64):
        print("{0} all: prop null - {1}, mean - {2}, median - {3}, max - {4},␣
↪min - {5}".format(col, ads[col].isna().sum() / 4268, ads[col].mean(),␣
↪ads[col].median(), ads[col].max(), ads[col].min()))
        print("{0} null: prop null - {1}, mean - {2}, median - {3}, max - {4},␣
↪min - {5}".format(col, no_end[col].isna().sum() / no_end.shape[0],␣
↪no_end[col].mean(), no_end[col].median(), no_end[col].max(), no_end[col].
↪min()))

    # if the column doesn't have quantitative data, just calculate the␣
↪proportion missing
    else:
        print("{0} all: prop null - {1}".format(col, ads[col].isna().sum() /␣
↪4268))
```

```
ADID all: prop null - 0.0
CreativeUrl all: prop null - 0.0
Currency Code all: prop null - 0.0
Spend all: prop null - 0.0, mean - 1673.9714151827554, median - 197.5, max -
451244, min - 0
Spend null: prop null - 0.0, mean - 1772.718226068145, median - 206.0, max -
451244, min - 0
Impressions all: prop null - 0.0, mean - 702295.6237113402, median - 72689.5,
max - 234901755, min - 1
Impressions null: prop null - 0.0, mean - 755473.5740941049, median - 80482.0,
max - 234901755, min - 1
StartDate all: prop null - 0.0
EndDate all: prop null - 0.18205248359887535
OrganizationName all: prop null - 0.0
BillingAddress all: prop null - 0.0
CandidateBallotInformation all: prop null - 0.866447985004686
PayingAdvertiserName all: prop null - 0.0
Gender all: prop null - 0.9097938144329897
AgeBracket all: prop null - 0.08270852858481724
CountryCode all: prop null - 0.0
Regions (Included) all: prop null - 0.7242268041237113
Regions (Excluded) all: prop null - 0.989456419868791
Electoral Districts (Included) all: prop null - 0.9847703842549204
Electoral Districts (Excluded) all: prop null - 1.0, mean - nan, median - nan,
max - nan, min - nan
Electoral Districts (Excluded) null: prop null - 1.0, mean - nan, median - nan,
max - nan, min - nan
Radius Targeting (Included) all: prop null - 0.9283036551077788
Radius Targeting (Excluded) all: prop null - 0.9971883786316776
Metros (Included) all: prop null - 0.9587628865979382
Metros (Excluded) all: prop null - 0.9960168697282099
Postal Codes (Included) all: prop null - 0.8024835988753515
Postal Codes (Excluded) all: prop null - 0.9725866916588566
Location Categories (Included) all: prop null - 0.9962511715089035
Location Categories (Excluded) all: prop null - 0.9995313964386129
Interests all: prop null - 0.7436738519212746
OsType all: prop null - 0.9939081537019682
Segments all: prop null - 0.2935801312089972
Language all: prop null - 0.729381443298969
AdvancedDemographics all: prop null - 0.9744611059044048
Targeting Connection Type all: prop null - 1.0, mean - nan, median - nan, max -
nan, min - nan
Targeting Connection Type null: prop null - 1.0, mean - nan, median - nan, max -
nan, min - nan
Targeting Carrier (ISP) all: prop null - 1.0, mean - nan, median - nan, max -
nan, min - nan
Targeting Carrier (ISP) null: prop null - 1.0, mean - nan, median - nan, max -
nan, min - nan
```

```
CreativeProperties all: prop null - 0.1733833177132146

/opt/conda/lib/python3.7/site-packages/numpy/lib/nanfunctions.py:1112:
RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

### 2.2.1 First Assessment of Missingness

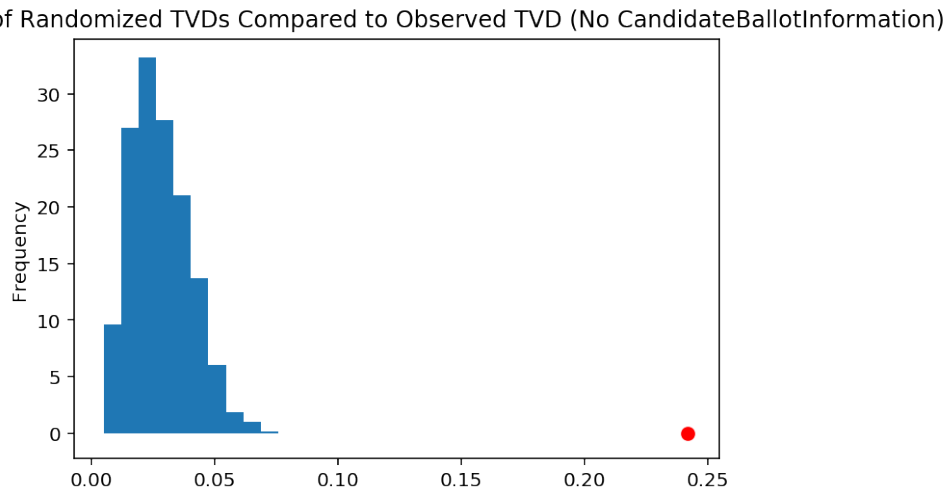
We compared the distributions of Null and Non-Null values for CandidateBallotInformation on Currency Code.
Test Statistic: Total Variation Distance
P-Value: 0.0

[32]:
```python
# defines total variation distance for later use
def tot_var_dist(dist1, dist2):
    return np.sum(np.abs(dist1 - dist2)) / 2
```

[33]:
```python
# specifies the columns for analyzing dependence
missing = 'CandidateBallotInformation'
depend = 'Currency Code'

# gets the proportions of currency codes for campaigns with missing␣
 ↪CandidateBallotInformation
no_end = ads[ads[missing].isna()]
no_end_dist = no_end.groupby(depend)['ADID'].count() / no_end['ADID'].count()

# gets the proportions of currency codes for campaigns with valid␣
 ↪CandidateBallotInformation values
all_ends = ads[ads[missing].notna()]
all_ends_dist = all_ends.groupby(depend)['ADID'].count() / all_ends['ADID'].
 ↪count()

# calculates observed tvd
observed = tot_var_dist(no_end_dist, all_ends_dist)

# running randomized tests
ads['missing_na'] = ads[missing].isna()
N =1000
tvds = []
for i in range(N):

    # shuffles CandidateBallotInformation null values
    shuffled_ends = (ads['missing_na'].sample(replace=False, frac=1).
 ↪reset_index(drop=True))
    shuffled = (ads[[depend]].reset_index(drop=True).assign(**{'Shuffled␣
 ↪Missing': shuffled_ends}))
```

```
    # gets the proportions of currency codes where shuffled␣
 ↪CandidateBallotInformation is null
    no_end = shuffled[shuffled['Shuffled Missing'] == False]
    no_end_dist = no_end.groupby(depend)['Shuffled Missing'].count() /␣
 ↪no_end['Shuffled Missing'].count()

    # gets the proportions of currency codes where shuffled␣
 ↪CandidateBallotInformation is not null
    all_ends = shuffled[shuffled['Shuffled Missing']==True]
    all_ends_dist = all_ends.groupby(depend)['Shuffled Missing'].count() /␣
 ↪all_ends['Shuffled Missing'].count()

    # calculates test tvd
    tvd = tot_var_dist(no_end_dist, all_ends_dist)
    tvds.append(tvd)

# plot tvd distribution
pd.Series(tvds).plot(kind='hist', density=True, title="Distribution of␣
 ↪Randomized TVDs Compared to Observed TVD (No CandidateBallotInformation)")
plt.scatter(observed, 0, color='red', s = 40);

# calculate p-value
p_val = np.count_nonzero(tvds >= observed)/ N
print("p-value: {}".format(p_val))
```

```
p-value: 0.0
```

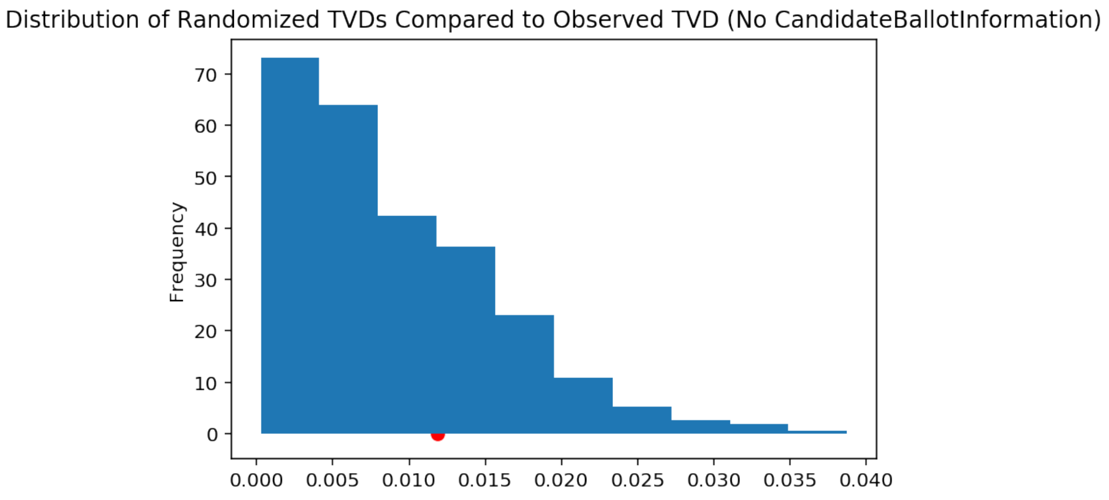


### 2.2.2 Second Assessment of Missingness

We compared the distributions of Null and Non-Null values for CandidateBallotInformation on Radius Targeting (Included).
Test Statistic: Total Variation Distance
P-Value: ~0.34

[34]:
```python
# specifies the columns for analyzing dependence
missing = 'CandidateBallotInformation'
depend = 'Radius Targeting (Included)'

# gets the data where CandidateBallotInformation is missing
val_missing = ads[ads[missing].isna()].copy()

# gets the proportions where Radius Targeting (Included) is and isn't missing
prop_null = val_missing[depend].isna().sum() / val_missing.shape[0]
prop_not_null = val_missing[depend].notna().sum() / val_missing.shape[0]
null_series = pd.Series([prop_null, prop_not_null])

# gets the data where CandidateBallotInformation isn't missing
val_missing = ads[ads[missing].notna()].copy()

# gets the proportions where Radius Targeting (Included) is and isn't missing
prop_null = val_missing[depend].isna().sum() / val_missing.shape[0]
prop_not_null = val_missing[depend].notna().sum() / val_missing.shape[0]
specified_series = pd.Series([prop_null, prop_not_null])

# calculate observed tvd
observed = tot_var_dist(null_series, specified_series)

# running randomized tests
N = 1000
results = []
relevant = ads.copy()
for _ in range(N):

    # shuffle CandidateBallotInformation
    shuffled_col = (
        relevant[depend]
        .sample(replace=False, frac=1)
        .reset_index(drop=True)
    )
    shuffled = (
        relevant
        .assign(**{
            'shuffled': shuffled_col,
        })
    )
    # gets the data where CandidateBallotInformation is missing
    val_missing = shuffled[shuffled[missing].isna()].copy()

    # gets the proportions where Radius Targeting (Included) is and isn't␣
↪missing
    prop_null = val_missing['shuffled'].isna().sum() / val_missing.shape[0]
    prop_not_null = val_missing['shuffled'].notna().sum() / val_missing.shape[0]
    null_series = pd.Series([prop_null, prop_not_null])

    # gets the data where CandidateBallotInformation isn't missing
    val_missing = shuffled[shuffled[missing].notna()].copy()

    # gets the proportions where Radius Targeting (Included) is and isn't␣
↪missing
    prop_null = val_missing['shuffled'].isna().sum() / val_missing.shape[0]
    prop_not_null = val_missing['shuffled'].notna().sum() / val_missing.shape[0]
    specified_series = pd.Series([prop_null, prop_not_null])

    # calculates test tvd
    results.append(tot_var_dist(null_series, specified_series))

# plot tvd distribution
pd.Series(results).plot(kind='hist', density=True, title="Distribution of␣
↪Randomized TVDs Compared to Observed TVD (No CandidateBallotInformation)")
plt.scatter(observed, 0, color='red', s = 40);

# calculate p-value
p_val = np.count_nonzero(results >= observed)/ N
print("p-value: {}".format(p_val))
```

```
p-value: 0.31
```



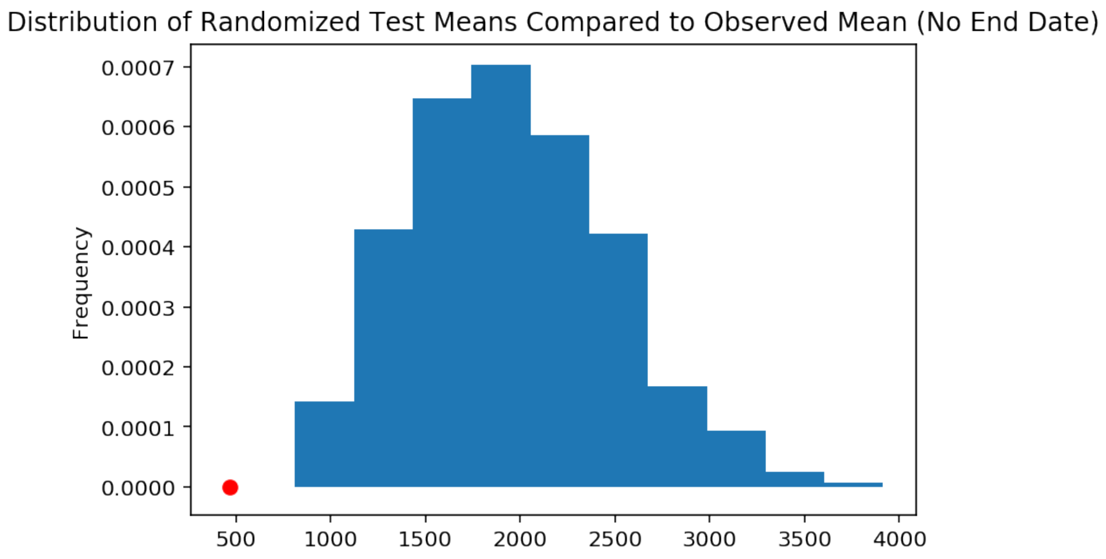
## 2.3 Hypothesis Test

**Question: for campaigns paid in USD, what is the difference between the average spend for campaigns where the end date is missing compared to the average spend for all campaigns?** Test statistic: Mean
P-Value: 0.0 - see the plot below for distribution

[35]:
```
# data where campaigns are paid in USD
all_ends = ads[ads['Currency Code'] == 'USD']
all_ends['end_na'] = all_ends['EndDate'].isna()

# data where the campaigns have no specified end date and are paid in USD
no_end = all_ends[all_ends['EndDate'].isna()]

# calculate observed statistic
observed = no_end['Spend'].mean()

# running randomized tests
N = 1000
means = []
for i in range(N):

    # shuffles the end column
    shuffled_ends = (all_ends['end_na'].sample(replace=False, frac=1).
 ↪reset_index(drop=True))
    shuffled = (all_ends[['Spend']].reset_index(drop=True).assign(**{'Shuffled␣
 ↪EndDate': shuffled_ends}))

    # calculates and saves the test statistic
    na_mean = (shuffled.groupby('Shuffled EndDate').agg({'Spend':'mean'}).
 ↪loc[True]['Spend'])
    means.append(na_mean)


# plot mean distribution
pd.Series(means).plot(kind='hist', density=True, title="Distribution of␣
 ↪Randomized Test Means Compared to Observed Mean (No End Date)")
plt.scatter(observed, 0, color='red', s = 40);

# calculate p-value
p_val = np.count_nonzero(means <= observed)/ N
print("p-value: {}".format(p_val))
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

```
p-value: 0.0
```



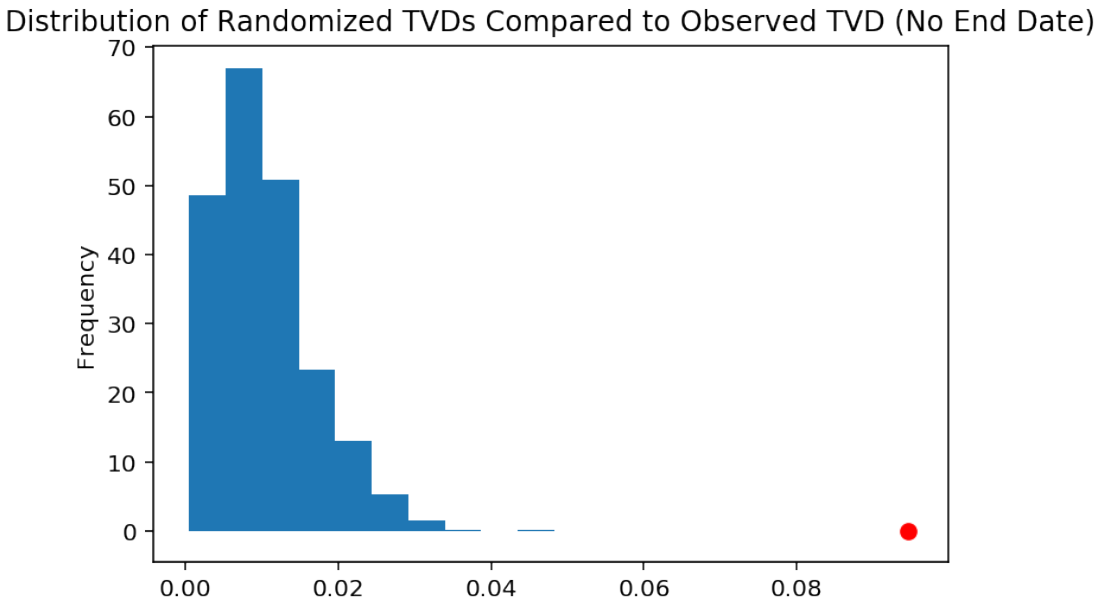
**Question: is there any difference between the distribution of specified gender-targeted campaigns where the campaign end date is missing compared to the distribution of specified gender-targeted campaigns for all campaigns?** (Used a permutation test)
Test statistic: Total Variation Distance
P-Value: 0.0 - see the plot below for distribution

[36]:
```python
# calculates gender proportions for campaigns without a specified end date
no_end = ads[ads['EndDate'].isna()]
no_end_dist = no_end.groupby('Gender')['ADID'].count() / no_end['ADID'].count()
dict_vers = no_end_dist.to_dict()
dict_vers['Null'] = no_end['Gender'].isna().sum()/no_end.shape[0]
no_end_dist = pd.Series(dict_vers)

# calculates gender proportions for campaigns with a specified end date
all_ends = ads[ads['EndDate'].notna()]
```

```
all_ends_dist = all_ends.groupby('Gender')['ADID'].count() / all_ends['ADID'].
 ↪count()
dict_vers = all_ends_dist.to_dict()
dict_vers['Null'] = all_ends['Gender'].isna().sum()/all_ends.shape[0]
all_ends_dist = pd.Series(dict_vers)

# calculates observed tvd
observed = tot_var_dist(no_end_dist, all_ends_dist)

# running randomized tests
ads['end_na'] = ads['EndDate'].isna()
N = 1000
tvds = []
for i in range(N):

    # shuffles the end column
    shuffled_ends = (ads['end_na'].sample(replace=False, frac=1).
 ↪reset_index(drop=True))
    shuffled = (ads[['Gender']].reset_index(drop=True).assign(**{'Shuffled␣
 ↪EndDate': shuffled_ends}))

    # calculates gender proportions for campaigns without a specified end date
    no_end = shuffled[shuffled['Shuffled EndDate'] == False]
    no_end_dist = no_end.groupby('Gender')['Shuffled EndDate'].count() /␣
 ↪no_end['Shuffled EndDate'].count()
    dict_vers = no_end_dist.to_dict()
    dict_vers['Null'] = no_end['Gender'].isna().sum()/no_end.shape[0]
    no_end_dist = pd.Series(dict_vers)

    # calculates gender proportions for campaigns with a specified end date
    all_ends = shuffled[shuffled['Shuffled EndDate']==True]
    all_ends_dist = all_ends.groupby('Gender')['Shuffled EndDate'].count() /␣
 ↪all_ends['Shuffled EndDate'].count()
    dict_vers = all_ends_dist.to_dict()
    dict_vers['Null'] = all_ends['Gender'].isna().sum()/all_ends.shape[0]
    all_ends_dist = pd.Series(dict_vers)

    # calculates test statistic
    tvd = tot_var_dist(no_end_dist, all_ends_dist)
    tvds.append(tvd)

# plot tvd distribution
pd.Series(tvds).plot(kind='hist', density=True, title="Distribution of␣
 ↪Randomized TVDs Compared to Observed TVD (No End Date)")
plt.scatter(observed, 0, color='red', s = 40);
```

```
# calculate p-value
p_val = np.count_nonzero(tvds >= observed)/ N
print("p-value: {}".format(p_val))
```

```
p-value: 0.0
```



```
[ ]:
```