STAT501-A2

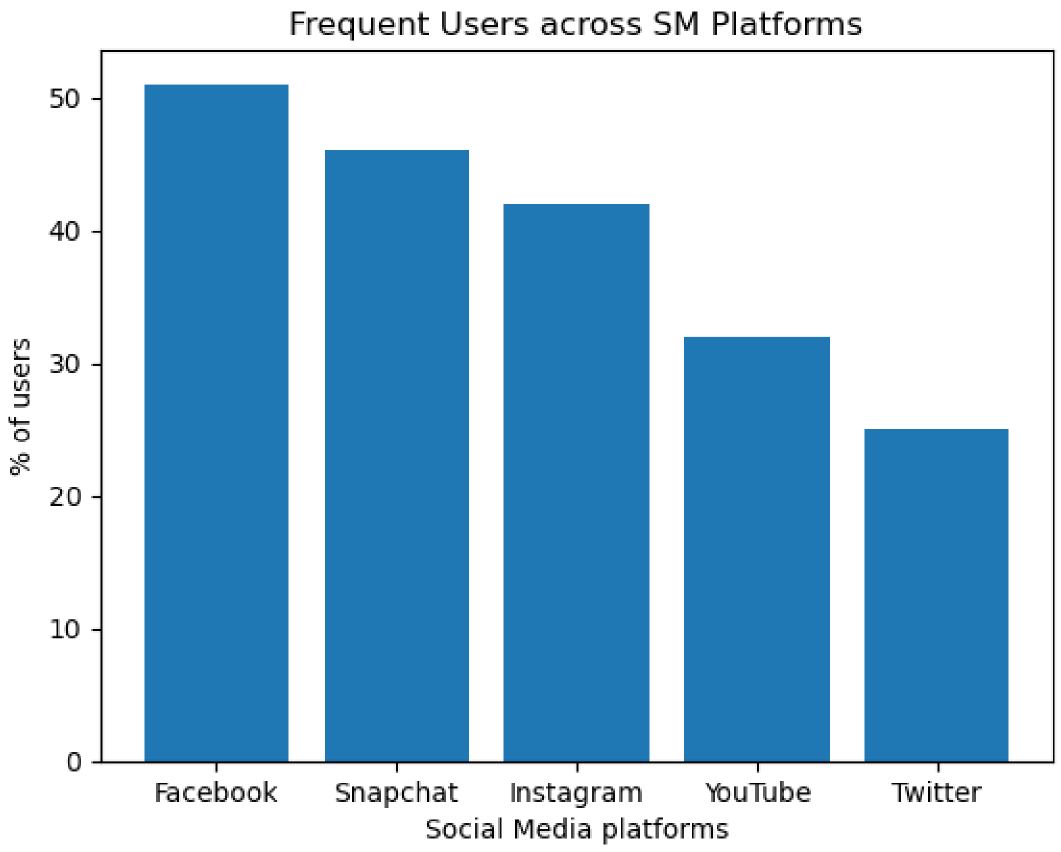
January 26, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.1 Exercise 1.12

[2]:
```
# read
data = pd.read_table('data/ex01-012socialm.txt', delimiter='\t')
```

[3]:
```
plt.bar(data['SocialMedia'], data['Users.pct.'])
plt.xlabel('Social Media platforms')
plt.ylabel('% of users')
plt.title('Frequent Users across SM Platforms')
plt.show()
```



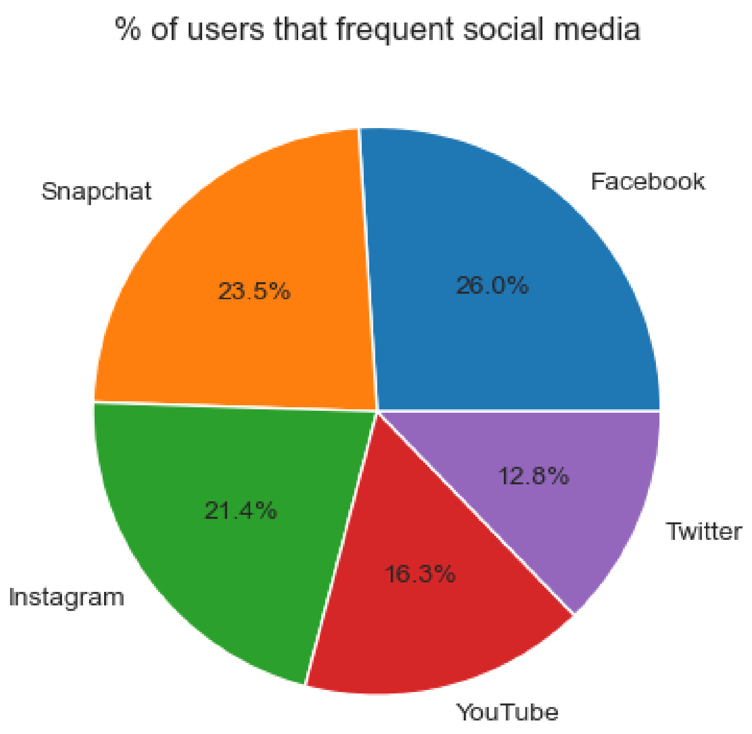
This chart conveys the following about the sample of users that use social media: - Approximately 50-51% of users frequent Facebook several times a day. - 46% of users frequent Snapchat. - 42% of users frequent Instagram. - 32% of users frequent YouTube. - 25% of users frequent Twitter.

Basically, the sample contains users that use multiple social media platforms and tend to visit them multiple times a day; some platforms more than others. The question of why may be down to user activity, engagement, or interests that are prevalent across platforms.

0.2 Exercise 1.13

0.2.1 a)

[4]:
```
sns.set_style("whitegrid")
plt.pie(data['Users.pct.'], labels=data['SocialMedia'], autopct='%1.1f%%')
plt.title('% of users that frequent social media')
plt.show()
```



From the pie chart, we have had to sum the values up (total is 196; where this equals 100%), and create a percentage of the total for each category. By this chart, people may interpret findings as: - 26% of users frequent Facebook - 23.5% of users frequent Snapchat - 21.4% of users frequent Instagram - 16.3% of users frequent YouTube - 12.8% of users frequent Twitter

This is misrepresenting the original values and does not show overlaps.

0.2.2 b)

The bar chart gives us more information on the data as it appears in tabular format. With the bar chart, you are able to infer that people tend to use multiple social media platforms; while this does not happen in the pie chart.

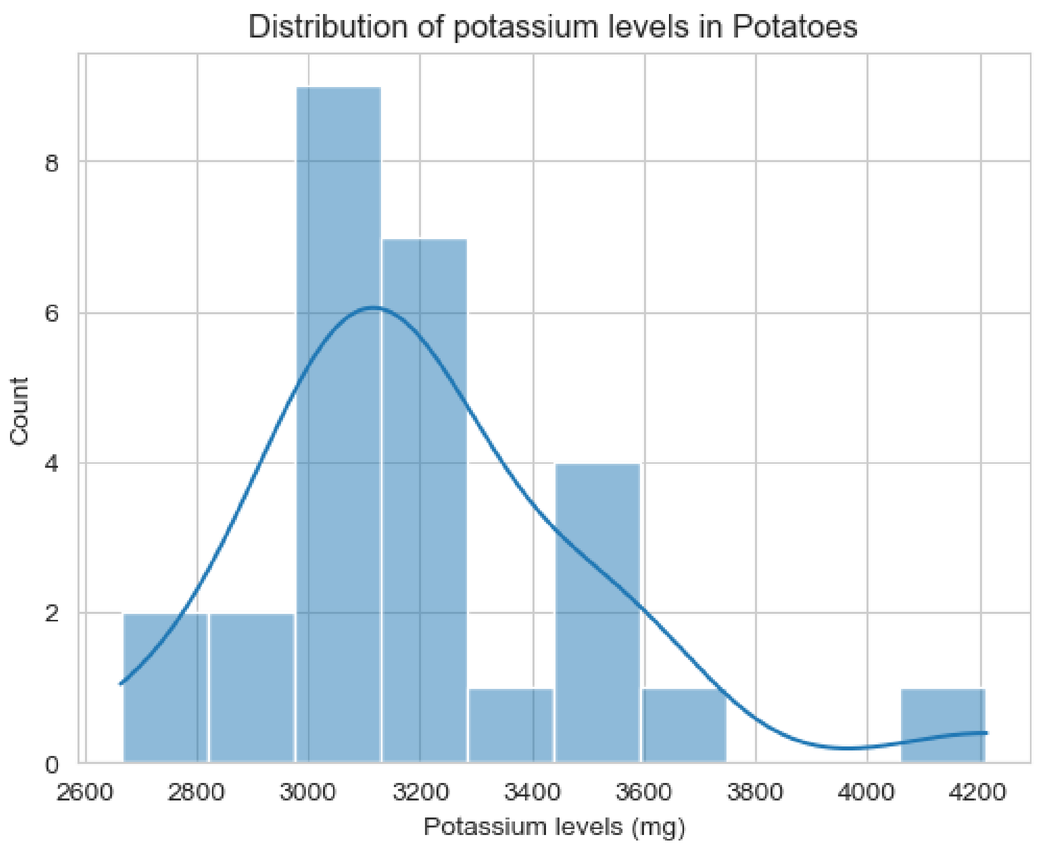
With a pie chart; the total of the values being examined need to be equal to 100; if they are not, they need to be scaled. This causes data to be misrepresented visually; even though the implication remains the same. Note the order of the platforms between the bar and pie charts - they are the same; yet different numerically.

0.3 Exercise 1.15

0.3.1 a)

[5]:
```python
data = pd.read_table('data/ex01-015kpot40.txt', delimiter= '\t')

# histogram
sns.histplot(data, x='Potassium_mg', kde=True)
plt.xlabel('Potassium levels (mg)')
plt.title('Distribution of potassium levels in Potatoes')
plt.show()
```



0.3.2 b)

The distribution is unimodal, and is positively skewed (right skewed). It is not symmetric. Potential outlier on the tail end.

0.3.3 c)

There are potential outliers between the 4050 - 4200 mark. The reason for this is because there are datapoints in this range that are too far out from the local maxima (just below 3800). That being said, I would not use this plot to identify outliers - but would instead use a boxplot with IQR (interquartile range). Any datapoints outside Q3 and Q1 are outliers.

0.3.4 d)

```
[6]: # get spread, sd, and median
     print(data['Potassium_mg'].describe())
```

```
count      27.000000
mean     3208.437407
std       306.676467
min      2664.380000
25%      3036.620000
50%      3130.370000
75%      3283.065000
max      4213.490000
Name: Potassium_mg, dtype: float64
```

As stated earlier - the shape of the plot is unimodal and asymmetric; right skewed.

The spread of values ranges between `2600 - 4200`. The center (median) is `3130.37`, and standard deviation of values is `306.67`. The sample mean is `3208.44`. Both the mean and median are close to each other; and since the data is right skewed - the mean is greater than the median. That being said, if we want to consider the suspected outlier in the dataset - the median is a more accurate measure of center.

Note: All values are rounded to 2 decimal places.

0.4 Exercise 1.29

0.4.1 a)

```
[7]: # Mean
     print('Mean:', data['Potassium_mg'].mean().round(2))
```

```
Mean: 3208.44
```

Mean is calculated in Python using the `numpy.mean()` library and function.

By hand, the mean is calculated using the equation: Sum of observations / Number of observations.

0.4.2 b)

```
[8]: # Median
     print('Median:', data['Potassium_mg'].median())
```

```
Median: 3130.37
```

Median is calculated the same way as the previous question, using the numpy library and the median function.

By hand, the median is found by sorting the values in ascending orders first, and counting the number of observations. If the number of observations is odd - the median is the middle value of all observations present. If even, then the middle value is the average of the two middle observations.

0.5 Exercise 1.31

0.5.1 b)

```
[9]: print('Q1:', data['Potassium_mg'].quantile([0.25]))
     print('Q2 / Midpoint:', data['Potassium_mg'].quantile([0.5]))
     print('Q3:', data['Potassium_mg'].quantile([0.75]))
```

```
Q1: 0.25    3036.62
Name: Potassium_mg, dtype: float64
Q2 / Midpoint: 0.5    3130.37
Name: Potassium_mg, dtype: float64
Q3: 0.75    3283.065
Name: Potassium_mg, dtype: float64
```

3 quartiles; - Q1: `3036.62` - Q2: `3130.37` - Q3: `3283.065`

Functions used (Python): `pandas.quantile()`

0.5.2 c)

```
[10]: print(data['Potassium_mg'].describe())
```

```
count      27.000000
mean     3208.437407
std       306.676467
min      2664.380000
25%      3036.620000
50%      3130.370000
75%      3283.065000
max      4213.490000
Name: Potassium_mg, dtype: float64
```

This is a little different than the five-number summary but still displays what we need.

- Min: `2664.38`
- Q1: `3036.62`
- Median / Q2: `3130.37`
- Q3: `3283.065`
- Max: `4213.49`

Obtained with the `pandas.describe()` function.

The first value is the minimum value of an observation in the dataset. The second is the first quartile, the third is the median or midpoint (Q2), the fourth is the third quartile, and the last is the maximum value of an observation in the dataset. A quartile or quantile is a group that divides the dataset into four parts of equal or near equal parts; based on distribution of the variable.

The min and max will tell us the spread (range) of the data - which is between `2664-4213`. The Q1 value tells us that 25% of the data falls below this quartile; the median is the midpoint, and also tells that 50% of the data falls below this quartile; and finally Q3 tells us that 75% of the data falls below this quartile. So if Q1 = 3036.62 - then any value below this falls within the first quartile.

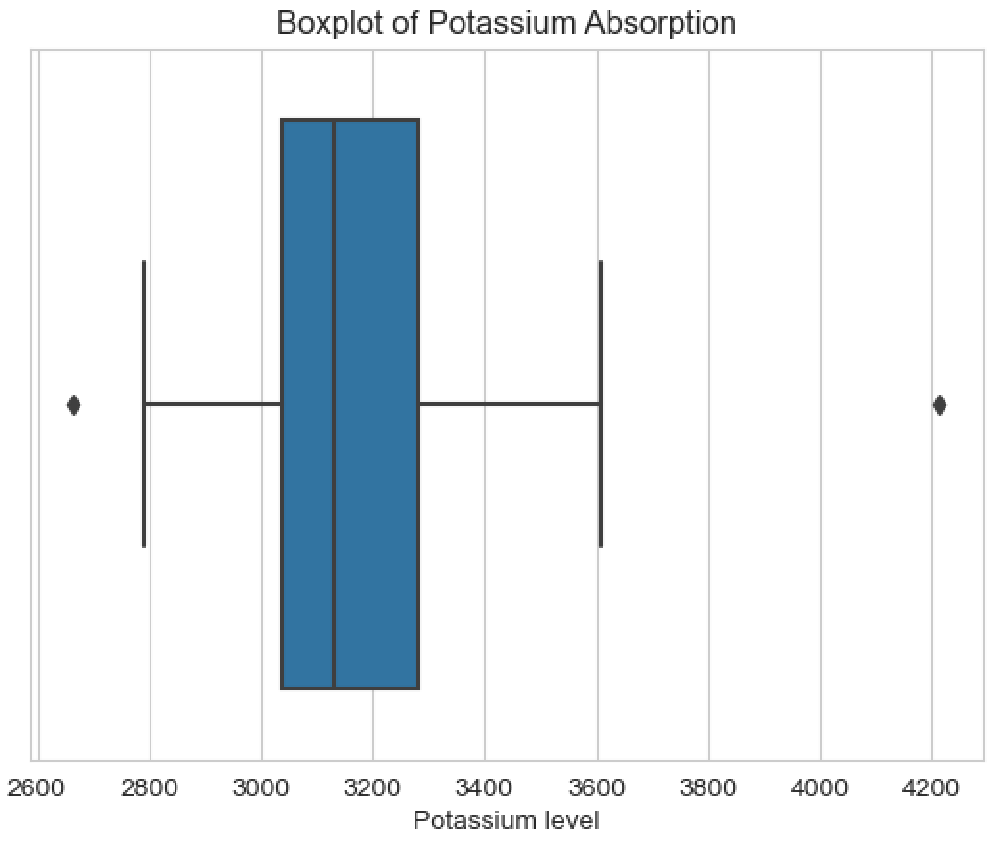
0.5.3 d)

The five number summary is the most useful in describing a distribution. Even without the mean, you will know where the center of the distribution lies, and estimate what it may look like without a graphical representation. The mean and standard deviation alone will not help us in describing a distribution.

0.6 Exercise 1.33

0.6.1 b)

[11]:
```python
sns.boxplot(data, x='Potassium_mg')
plt.xlabel('Potassium level')
plt.title('Boxplot of Potassium Absorption')
plt.show()
```



From this plot, we know that the distribution is right skewed (positively skewed). There are two outliers present in the dataset, according to this plot.

0.6.2 c)

This would depend on what I need to do. I do prefer the boxplot, since it makes it easier for me to find outliers while telling me what skew the data has. During workflows however, I tend to use both a histogram and a boxplot in conjunction. A histogram is easier to read - in that you can determine the spread and skew quite easily. A boxplot makes this a bit harder to do, but gives me details that a histogram does not; like the quartiles, median and outliers.

0.7 Exercise 1.37

0.7.1 a)

[12]:
```
data = pd.read_table('data/ex01-037stout.txt', delimiter='\t')

# drop the 'RelativeToPast4' column
data.drop('RelativeToPast4', axis=1, inplace=True)
```

[13]:
```
# calculate mean
print('Mean:', data['RelativeTo1913'].mean())
```

```
Mean: 122.91666666666667
```

Used the `numpy.mean()` function in Python to calculate the mean.

The mean is `122.92`; rounded to 2 decimal places.

0.7.2 b)

[14]:
```
# calculate the median
print('Median:', data['RelativeTo1913'].median())
```

```
Median: 102.5
```

Used the `numpy.median()` function in Python to retrieve the median.

The median is `102.5`.

0.8 Exercise 1.38

0.8.1 a)

[15]:
```
# calculate standard deviation of the data
print('Standard Deviation:', data['RelativeTo1913'].std())
```

```
Standard Deviation: 105.72817490951486
```

Used the `numpy.std()` function in Python to retrieve standard deviation.

SD is computed as `105.73`, rounded to 2 decimal places.

0.8.2 b)

```
[16]: # calculate quartiles
print(data['RelativeTo1913'].quantile([0.25, 0.5, 0.75]))
```

```
0.25     67.5
0.50    102.5
0.75    123.5
Name: RelativeTo1913, dtype: float64
```

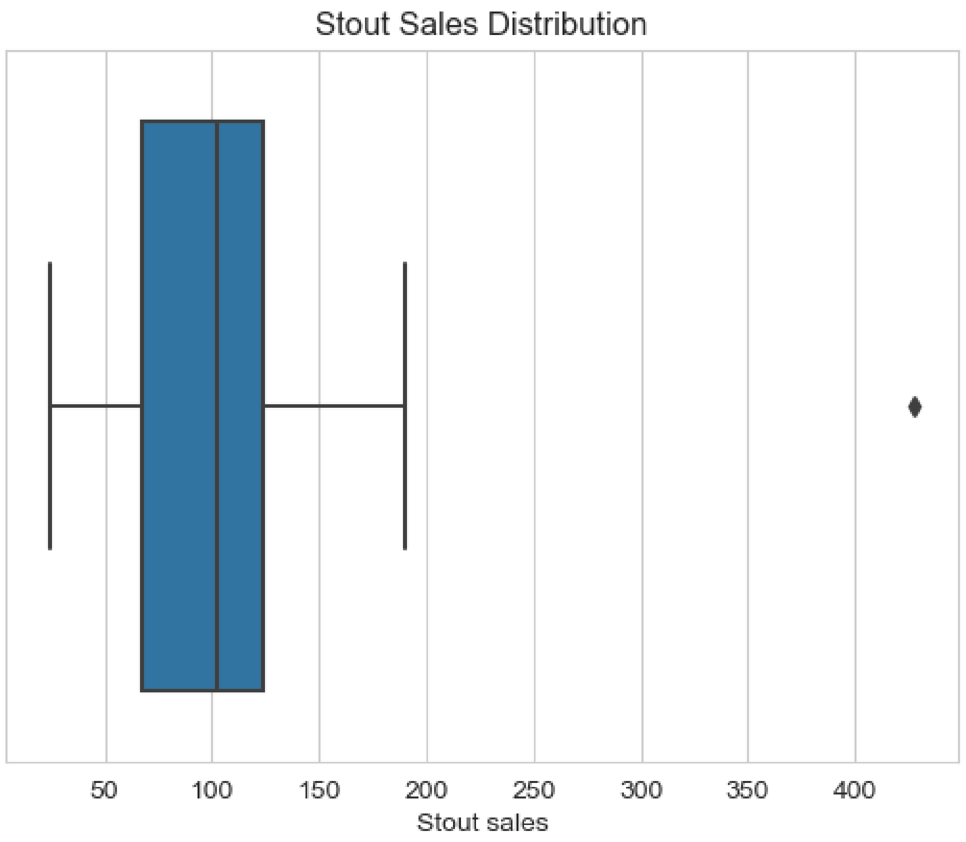
Used the `pandas.quantile()` function in Python to retrieve Q1, Q2, and Q3. The quantiles are computed as: - Q1: `67.5` - Q2: `102.5` - Q3: `123.5`

0.8.3 c)

```
[17]: data.describe()
```

```
[17]:        RelativeTo1913
count        12.000000
mean        122.916667
std         105.728175
min          24.000000
25%          67.500000
50%         102.500000
75%         123.500000
max         428.000000
```

```
[18]: sns.boxplot(data, x='RelativeTo1913')
plt.xlabel('Stout sales')
plt.title('Stout Sales Distribution')
plt.show()
```



Given that we know that there are outliers, a box plot is preferable. The spread or range of the data is between `24-428`. The distribution itself is left skewed, and there is a potential outlier in the dataset. Combining this from the summary we obtained above, we have an idea of what the spread is like.

I am using a modified boxplot from 1.39 with the help of the `seaborn.boxplot()` function.

0.9 Exercise 1.39

0.9.1 a)

[19]:
```python
def get_IQR(data):
    Q1, Q3 = data.quantile(0.25), data.quantile(0.75)
    IQR = Q3-Q1
    return IQR

iqr = get_IQR(data['RelativeTo1913'])
print('The Inter-Quartile Range for the data is:', iqr)
```

```
The Inter-Quartile Range for the data is: 56.0
```

I created a custom function that computes the interquartile range using the `pandas.quantile()`

function. The interquartile range is computed by subtracting Q1 from Q3 (see function), and returned.

The IQR is computed as 56.

0.9.2 b)

```
[20]: def get_outliers(data, column_name):
          Q1, Q3 = data[column_name].quantile(0.25), data[column_name].quantile(0.75)
          IQR = Q3-Q1
          lower_threshold = Q1-1.5*IQR
          upper_threshold = Q3+1.5*IQR
          outlier = data[(data[column_name] < lower_threshold) | (data[column_name] >␣
        ↪upper_threshold)]
          return outlier

      outliers = get_outliers(data, 'RelativeTo1913')
      print(outliers)
```

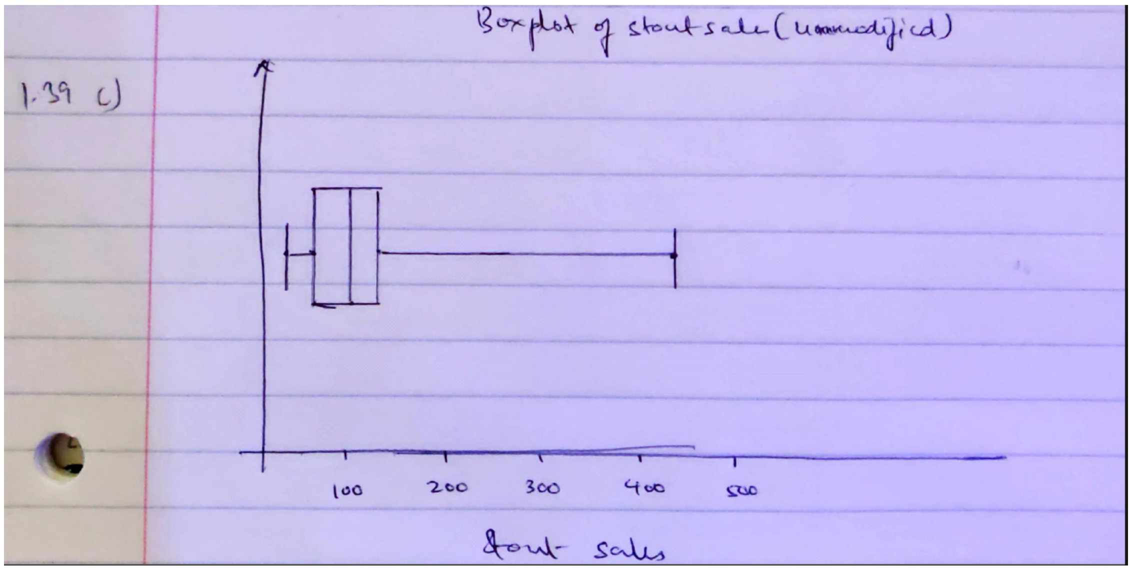
```
   Region  RelativeTo1913
8  London             428
```

One outlier was identified; namely the record for London that has a value of 428.

I created a custom function based on the previous one for IQR - and just extended it to add an upper and lower limit using the 1.5*IQR rule. This is returned as a filtered record via the original dataframe, so that we know which datapoint is an outlier.

We know this is not a suspected outlier (because this is now a confirmed outlier), as the datapoint is quite far away from other values, as seen in the boxplot above.

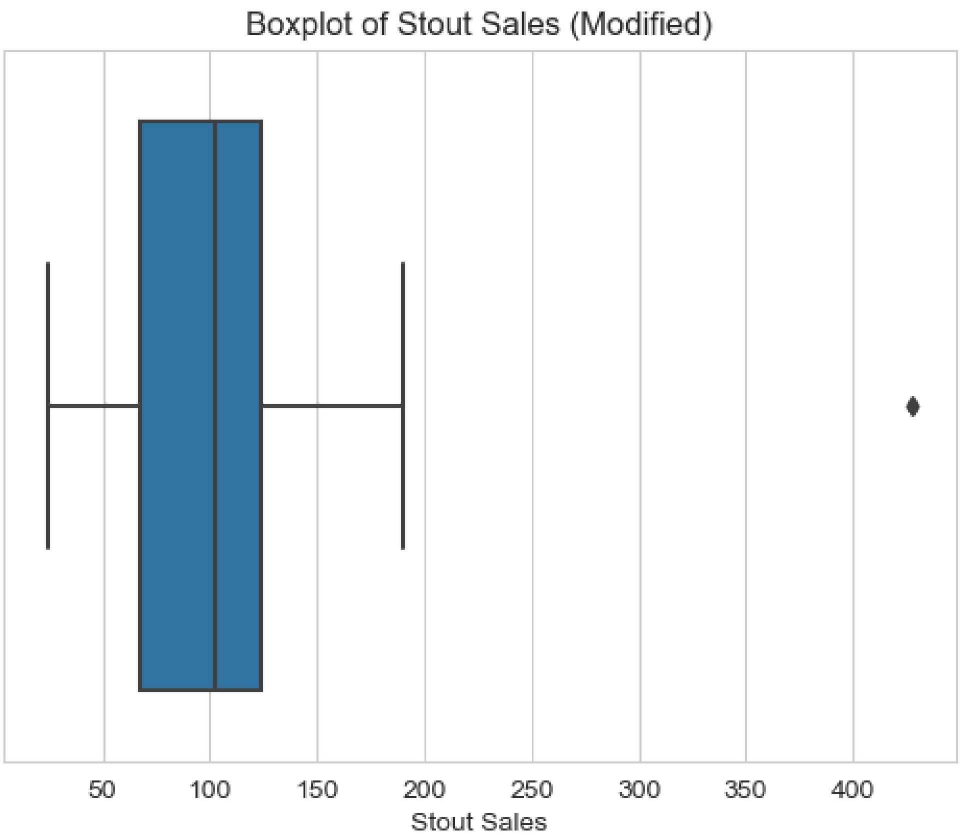
0.9.3 c)



This boxplot indicates that the dataset is negatively skewed (left skewed). The median is just past the 100 value, which is consistent with our earlier finding of `102.5`. As this is an unmodified boxplot, we cannot see outliers outright, but can infer from the image. In this case, anything past Q3 can be considered a suspected outlier.

0.9.4 d)

[22]:
```python
sns.boxplot(data, x='RelativeTo1913')
plt.xlabel('Stout Sales')
plt.title('Boxplot of Stout Sales (Modified)')
plt.show()
```



0.9.5 f)

Comparing the two, they are largely the same, except in how they look on the right side of the distribution. The unmodified boxplot does not identify suspected outliers, so the whisker extends all the way to the maximum value in the dataset. In the modified boxplot, which does identify suspected outliers, the whisker is pulled back to the last value that is not identified as an outlier according to the 1.5-IQR rule. The distibution however, remains the same.

The record we found in b) is an outlier, however - this outlier carries a lot of weight when context from the problem is applied to it. It is not far out of the realm of possibility that alcohol consumption would have increased just before the outbreak of war; especially in a large city like London, where alcohol sales will be higher than othe cities.

0.10 Exercise 1.43

0.10.1 a)

[23]:
```
data = pd.read_table('data/ex01-043potato.txt', delimiter='\t')

# summarize data
print(data.describe())
```

```
          weight
count  25.000000
mean    6.456000
std     1.424804
min     3.700000
25%     5.200000
50%     6.700000
75%     7.800000
max     8.200000
```

[24]:
```
# get IQR
print('The IQR is:', get_IQR(data['weight']).round(3))
```

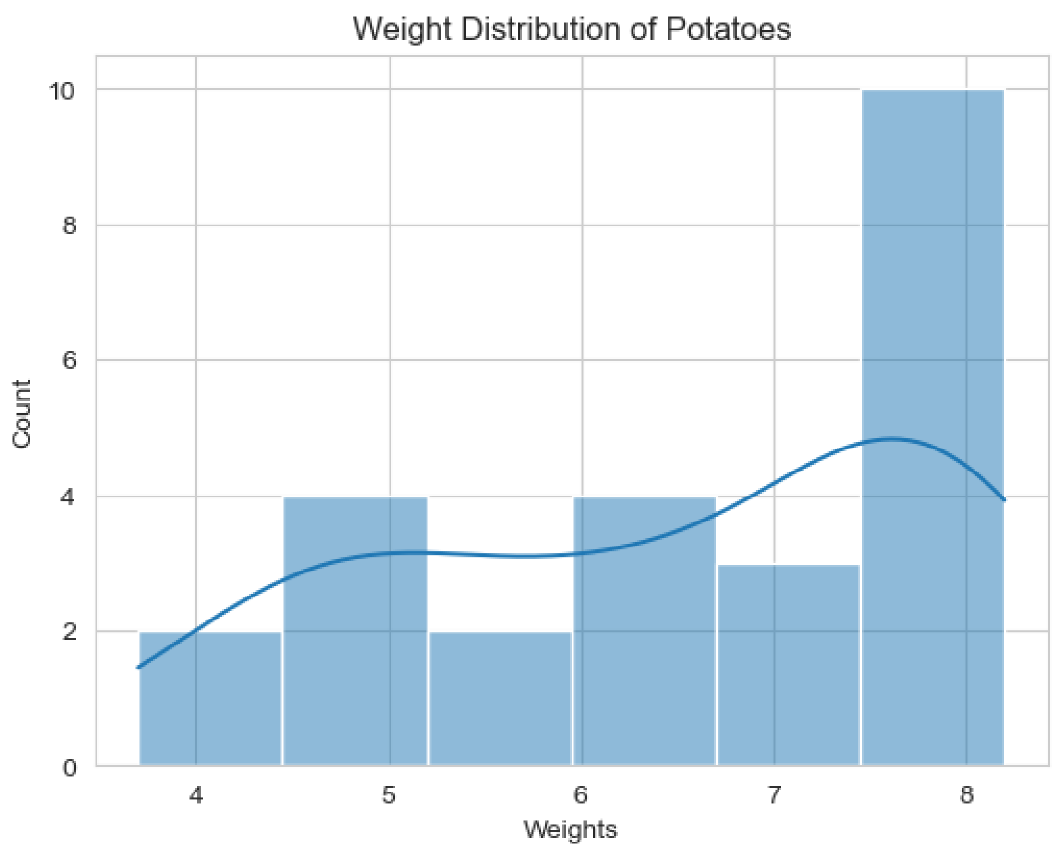
```
The IQR is: 2.6
```

[25]:
```
# Mode?
print(data['weight'].mode())
```

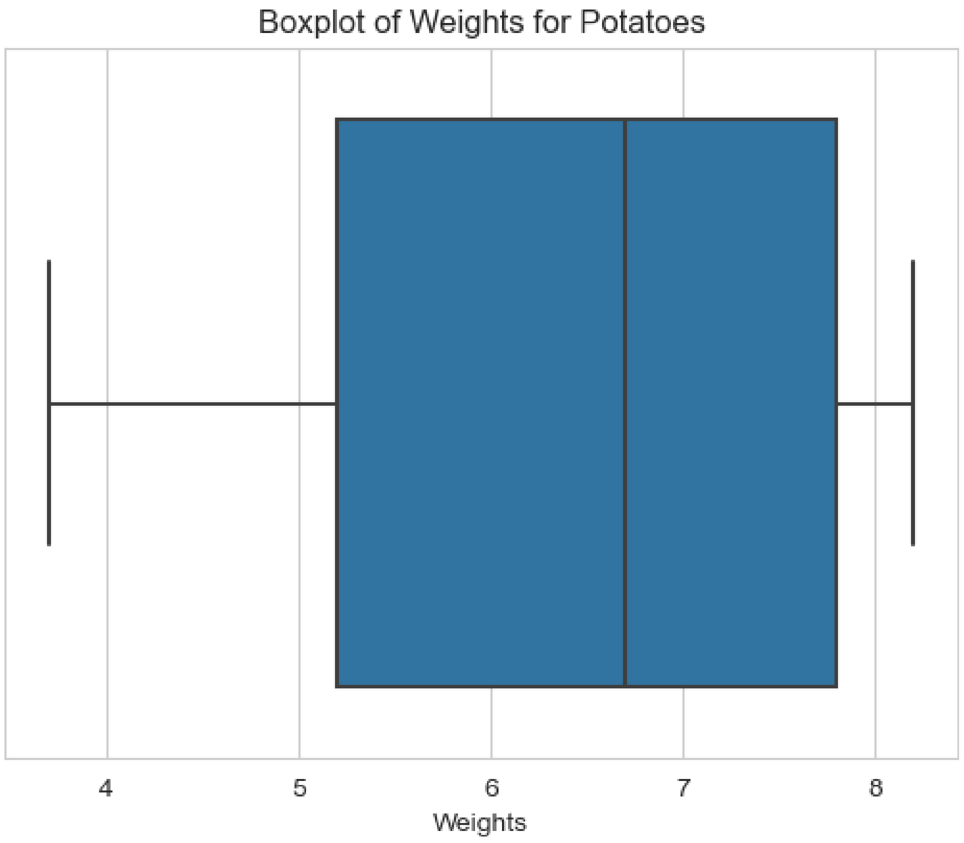
```
0    7.9
Name: weight, dtype: float64
```

Used the `numpy.mode()` function to retrieve the mode of the single variable provided.

[26]:
```
sns.histplot(data, x='weight', kde=True)
plt.xlabel('Weights')
plt.title('Weight Distribution of Potatoes')
plt.show()
```



[27]:
```
sns.boxplot(data, x='weight')
plt.xlabel('Weights')
plt.title('Boxplot of Weights for Potatoes')
plt.show()
```



Graphically, the data is bimodal and asymmetric. More than half of the data present falls under Q2. The data appears to be left skewed. It appears that most of our values are further away from the minimum.

Numerically, the mean is `6.456`, and the median is `6.7` which is quite close to each other. The lower quartile (Q2) value is `5.2`, while the upper quartile value is `7.8`. The range of the data is in between `3.7 - 8.2`, and the interquartile range is `2.59 (2.6)`.

I have used the 5 number (in this case, 8 number) summary to determine the spread of the distribution, its standard deviation, mean and median for a measure of center. I have also used the custom function I created to determine the interquartile range as it gives us further insight into spread. In addition to numeric descriptions - I also used two graphical descriptions, a histogram and a boxplot to describe the dataset. The histogram tells us that the data is bimodal, and the boxplot tells us how the data is skewed, while also showing us where the 5 number summaries are located when visualized.

0.10.2 b)

No. I do not think a numerical description, without a graphical representation is effective as a numeric description does not capture the fact that the data we have is bimodal. If we were asked to state what the center of the distribution was - it would be easy to assume that either the median or mean was a good estimate of centrality. I am of the opinion that both numeric and graphical descriptions are essential to conveying information to a researcher.

0.10.3 c)

I don't think so. For one - the data we have has too few observations to warrant a split; the second thing is that splitting into subsamples may not yield any significant results. Consider the spread of the data; if we split without replacement using random sampling into equivalent subsets of 12 each (this is technically not equal since the total size of the sample is 25) - we will see changes in the mean and median but the standard deviation will largely show an insignificant change. This is not a statistically significant finding.

0.11 Exercise 1.111

0.11.1 a)

A bar chart for both would convey all information needed. However, if a percentage is needed for how old the cars are - a pie chart might be a consideration; however, I tend to stay away from pie charts.

0.11.2 b)

Again, a bar chart is best for this. A stacked bar chart if we have information for two semesters so that the difference between both is conveyed in one diagram. If time series data is available, a line chart that shows us change between both semesters.

0.11.3 c)

Again, a bar chart works here. Filter by top 5 or 3 and this should be enough.