# Statistics Homework 03

B08705034 資管二 施芊羽

## Chapter 4

### 4.3

**a.**

In [1]:
```
from matplotlib import pyplot as plt
%matplotlib inline
# 設定圖形大小; DPI越大圖越大
plt.rcParams["figure.dpi"] = 150
import seaborn as sns
import pandas as pd
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf
import math as math
import statistics
```

In [2]:
```
data = [5.5, 7.2, 1.6, 22.0, 8.7, 2.8, 5.3, 3.4, 12.5, 18.6, 8.3, 6
.6]

print("Mean: ", np.mean(data), "miles")
print("Median: ", np.median(data), "miles")
print("Mode: ", stats.mode(data))
```

```
Mean:  8.541666666666666 miles
Median:  6.9 miles
Mode:  ModeResult(mode=array([1.6]), count=array([1]))
```

| Mean | Median | Mode |
|---|---|---|
| $8.54$ | $6.9$ | All the elements |

Unit: mile(s)

**b.**

From the given data, mean and median we solved, we can know that the mean of the numbers is larger than the median. This means that there are some greater numbers which increase the mean while the median is much smaller. From the mode, we can know that all numbers occur in the same frequency.

## 4.11

**a.**

In [8]:
```
price = [12, 10, 14, 15, 22, 30, 25]
rate = np.zeros(6)
for i in range(6):
    print("The rate of year ", i + 1, " is ", round((price[i + 1] / price[i])*100, 2), "%")
    rate[i] = round((price[i + 1] / price[i])*100, 2)
```

```
The rate of year  1  is  83.33 %
The rate of year  2  is  140.0 %
The rate of year  3  is  107.14 %
The rate of year  4  is  146.67 %
The rate of year  5  is  136.36 %
The rate of year  6  is  83.33 %
```

$$\frac{price[i+1] - price[i]}{price[i]}$$

**b.**

In [9]:
```
print("Mean: ", round(np.mean(rate), 2), "%")
print("Median: ", np.median(rate), "%")
```

```
Mean:  116.14 %
Median:  121.75 %
```

**c.**

In [10]:
```
def geomean(rate):
    r = rate
    geo_m = np.round(math.exp(np.log(r).mean()), 2)
    return geo_m

avg_rate = geomean(rate)
print("Geometric Mean =", avg_rate, " %")
```

```
Geometric Mean = 113.01  %
```

**d.**

It's best to use geometric mean to describe the given data since the rate of return will be multiply year after year, if we only use mean or median, we cannot count in the rate that being timed in the past years, which cannot correctly describe the real fact. If we use the mean $116.14\%$, after calculating, the price should be $29.449$ which is far higher than $25$ and the median is even higher which will be far more uncorrect. Therefore, when we're calculating the rate of something, which includes multiplication, it'll be better to use geometric mean.

## 4.17

**a.**

In [4]:
```
df_c04_17 = pd.read_excel("Xr04-17.xlsx")
print("Mean = ", np.round(df_c04_17["Speeds"].mean(), 2), "km/hr")
print("Median = ", np.round(df_c04_17["Speeds"].median(),2), "km/hr
")
```

```
Mean =  32.91 km/hr
Median =  32.0 km/hr
```

After the program running, we can get:

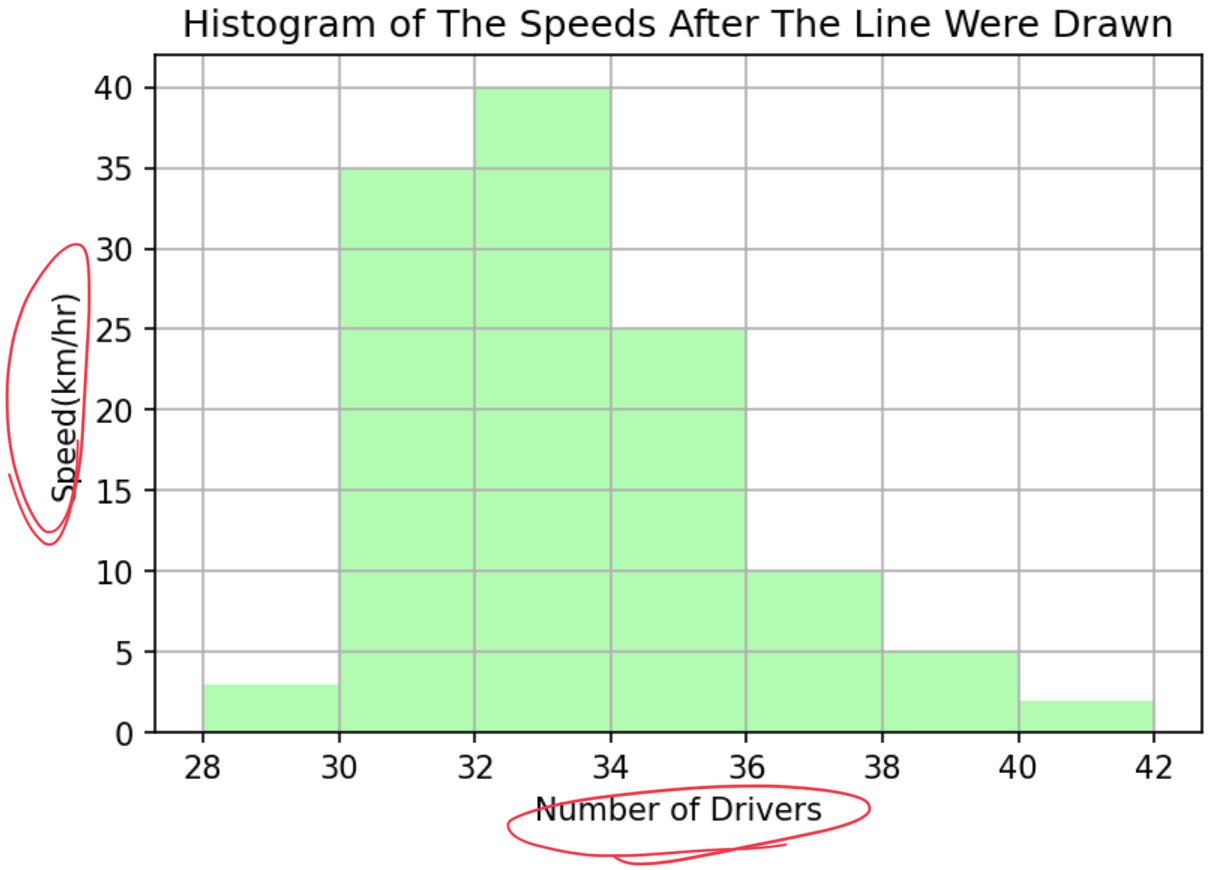
| Mean | Median |
|---|---|
| 32.91 | 32.0 |

Unit: $\frac{km}{hr}$

**b.**

In [13]:
```
bins_list = [28, 30, 32, 34, 36, 38, 40, 42]
fig, ax = plt.subplots()
counts, bins, patches = plt.hist(df_c04_17["Speeds"], bins=bins_lis
t, density=False, facecolor='palegreen', alpha=0.75)
plt.xlabel('Number of Drivers')
plt.ylabel('Speed(km/hr)')
plt.title('Histogram of The Speeds After The Line Were Drawn')
plt.grid(True)
plt.xticks(bins_list)

plt.show()
```




After the calculation in (a), we have known that the median of the data is smaller than the mean of the given data $\Rightarrow 32 \lt 32.91$(kilometers per hour), then we can know that the histogram of the given data will be postively skewed. Hence, I use `python` to draw a histogram to justify. We can say that there are half of the car speeds below or equal to $32$ (kilometers per hour) while the other half must be greater than or equal to $32$(kilometers per hour) since the median is $32$(kilometers per hour).

## 4.31

I do think that the **b.** sample has the smallest amount of variation while the **c.** has the greatest. I get my prediction from observation, I take a look at the given numbers and find out that set b. has the smallest range, and c. has the largest as $6$ and $39$ both exist in the sample.

In [14]:
```
a = [17, 29, 12, 16, 11]
b = [22, 18, 23, 20, 17]
c = [24, 37, 6 , 39, 29]
print("The variation of sample a: ", np.round(statistics.variance(a
), 2))
print("The variation of sample b: ", np.round(statistics.variance(b
), 2))
print("The variation of sample c: ", np.round(statistics.variance(c
), 2))
```

```
The variation of sample a:  51.5
The variation of sample b:  6.5
The variation of sample c:  174.5
```

Even though the exercise description doesn't ask me to get the variations of the samples, I still use `pstdev()` function in `statistics` to calculate them. As the result shows, my prediction is correct.

## 4.41

**a.**

In [39]:
```
df_c04_41 = pd.read_excel("Xr04-41.xlsx")

print("The variation of sample a: ", np.round(statistics.variance(df_c04_41["Punter 1"]), 2), "\nThe standard deviation of sample a: ", np.round(statistics.stdev(df_c04_41["Punter 1"]), 2))
print("The variation of sample b: ", np.round(statistics.variance(df_c04_41["Punter 2"]), 2), "\nThe standard deviation of sample b: ", np.round(statistics.stdev(df_c04_41["Punter 2"]), 2))
print("The variation of sample c: ", np.round(statistics.variance(df_c04_41["Punter 3"]), 2), "\nThe standard deviation of sample c: ", np.round(statistics.stdev(df_c04_41["Punter 3"]), 2))

df_c04_41.describe()
```

```
The variation of sample a:  40.22
The standard deviation of sample a:  6.34
The variation of sample b:  14.81
The standard deviation of sample b:  3.85
The variation of sample c:  3.63
The standard deviation of sample c:  1.91
```

Out[39]:

| | Punter 1 | Punter 2 | Punter 3 |
|---|---|---|---|
| **count** | 50.000000 | 50.000000 | 50.000000 |
| **mean** | 39.220000 | 39.360000 | 40.140000 |
| **std** | 6.341602 | 3.847925 | 1.906059 |
| **min** | 28.000000 | 30.000000 | 37.000000 |
| **25%** | 34.000000 | 36.000000 | 39.000000 |
| **50%** | 40.000000 | 40.000000 | 40.000000 |
| **75%** | 43.750000 | 42.000000 | 41.000000 |
| **max** | 56.000000 | 47.000000 | 45.000000 |

After the program running, we can get:

| | Punter 1 | Punter 2 | Punter 3 |
|---|---|---|---|
| Variance | 40.22 | 14.81 | 3.63 |
| Standard Deviation | 6.34 | 3.85 | 1.91 |

Unit: Variance = $miles^2$, Standard Variation = $mile(s)$

**b.**

From the statistics, we can know that the Punter 3 might be the most stable and strong one which can punt the ball with a farther average distance and have a smallest standard deviation. The Punter 1 may be the most unstable one since it has the highest std. As a coach. I'll choose the Punter 3 as the punter of our team from the statistics, while the Punter 1 is the last one to choose from the three.

## 4.47

In [37]:
```
df_c04_47 = pd.read_excel("Xr04-47.xlsx")

df_c04_47.describe()
```

Out[37]:

| | Property tax |
|---|---|
| **count** | 210.000000 |
| **mean** | 1937.316286 |
| **std** | 949.990956 |
| **min** | 221.330000 |
| **25%** | 1403.765000 |
| **50%** | 1741.745000 |
| **75%** | 2156.890000 |
| **max** | 5794.000000 |

After the program running, we can get:

| Mean | Standard Derivation |
|---|---|
| 1937.32 | 949.99 |

Unit: Mean = $dollars$, Standard Derivation = $dollar$.

1. The results are rounded to the two decimal places.

Since the amounts are highly positively skewed then we can also prove it from the mean is greater than the median. Moreover, since it's highly skewed, it means that the standard deviation might be a bit large. We can also conclude that the difference between median and max will be far more larger than the difference between median and min since it's positively skewed.