Descriptive Statistics In Python

What it is

Descriptive statistics is a branch of statistics that calculates and analyzes a given set of data, and uses it to provide information about the data you want to analyze.

We use statistics because:

1. We can understand the average value of any KPI (Key Performance Indicators) like Sales, Customer Count, NPS, Revenue, Market Basket, etc.
2. Mean, median, and mode are widely used imputations for missing values
3. These central tendencies (mean, median, mode) help us validate the data and align them with our knowledge, e.g. we understand that a grocery shop average sale volume increases during festive seasons when compared to the rest of the year.
4. When data is procured and average figures are observed, if the central tendencies are significantly different from what is known, we need to check the data quality
5. Linear regression, logistic regression, and neural networks follow certain assumptions which are validated using correlation, outlier checks, and so on

Terms in Statistics:

Population — The universe of all possible data for a given scenario (but it is rare that a population is observed because of data leaks or losses)

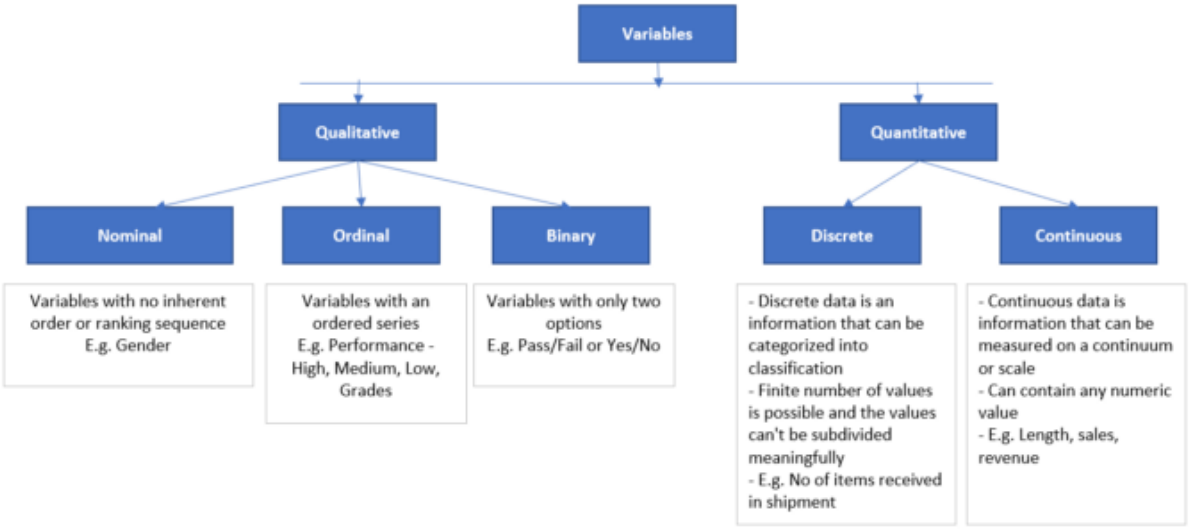
Sample — A set or filter of observations from a given population

Parameter — A numerical summary or value associated with a POPULATION. E.g. Average debt of all customers with loan accounts in a bank

Statistics — A numerical summary or value associated with a SAMPLE. E.g. Average calls made by customers opting for a customer care facility at the start of the week

Data Types

There are qualitative and quantitative data observations, and subsets of those, shown below:



Basic Functions

Before we get started, here are some basic functions that help with graphing, importing data, and using Python for statistics (the example file here is a calendar relating to sales):

In [3]:
```
# these are required libraries that are used for statistics
import pandas as pd
import numpy as np

# this is the read file function from pandas
df=pd.read_csv('sample_calendar.csv')

# the head() function displays the 1st 5 rows of the file
df.head()
```

Out[3]:

	Unnamed: 0	fiscal_week	week_no	holiday_week	pre_holiday_week	post_holiday_week	holiday_na
0	0	20190101	2019-02-09	0	0	0	
1	1	20190102	2019-02-16	0	0	1	
2	2	20190103	2019-02-23	1	0	0	['Presid D
3	3	20190104	2019-03-02	0	1	0	
4	4	20190201	2019-03-09	0	0	0	

1. Measures of Central Tendency

These variables tend to measure the 'average' behavior of the data.

1a. Mean

It is the average value of data, or the *weighted center*.

Can be found by:

$$X = [x_1 \;\; x_2 \;\; \ldots \;\; x_N]$$

$$Mean = \frac{x_1 + x_2 + \cdots + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

OR:

In [ ]:
```
number_of_rows=len(x)
total=x.sum()
mean=x.sum()/len(x)
```

OR:

In [ ]:
```
mean=x.mean() # this function is from numpy
```

1b. Median

It is the middle positional value, or the *unweighted center*.

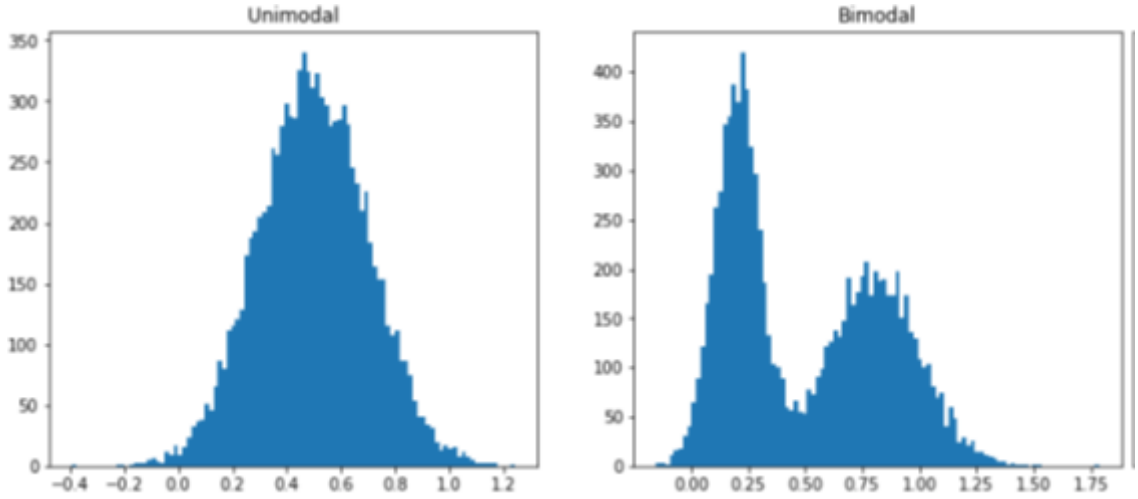
Found by:

In [ ]:
```
median=x.median() # this function is from numpy also
```

1c. Mode

The mode is the most frequent value in a set.

Sometimes, there are two modes (two values with high frequency), and this type of distribution is called a bimodal distribution, as shown below:



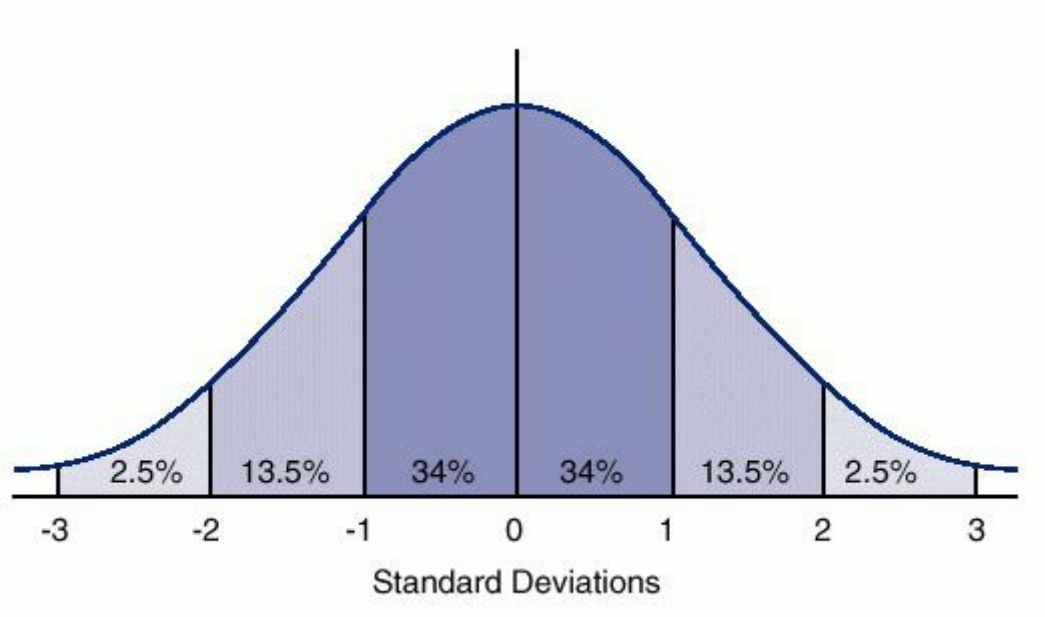
It can be found by:

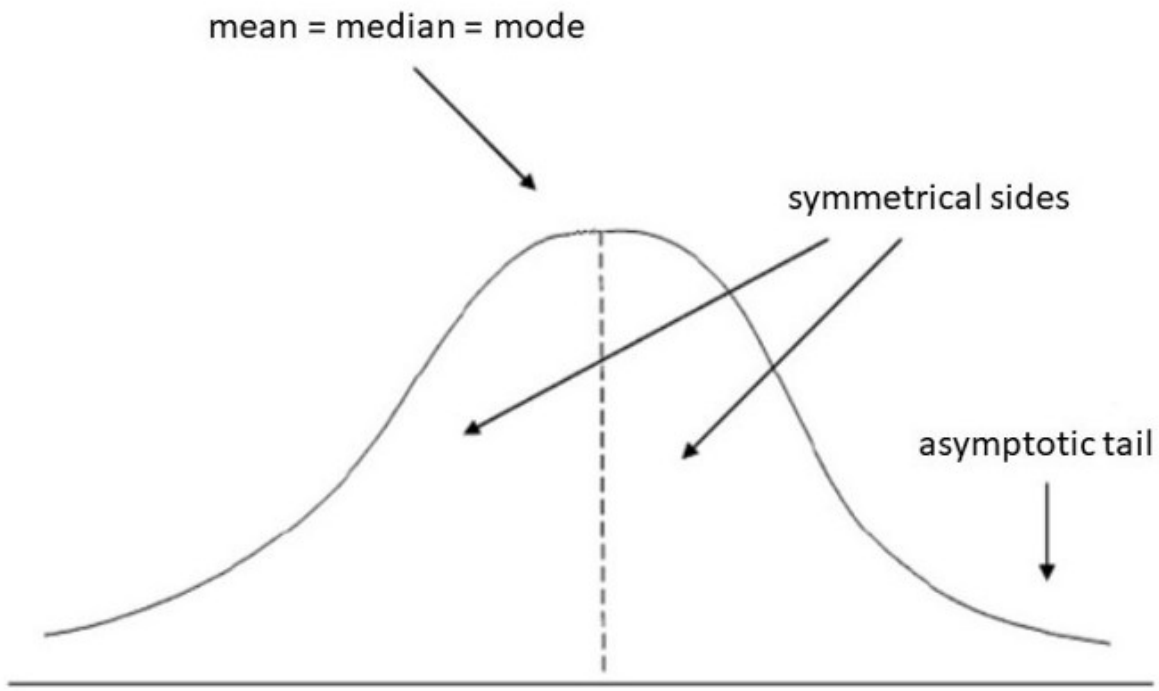
In [ ]:
```
mode=x.mode()
```

1d: Distributions

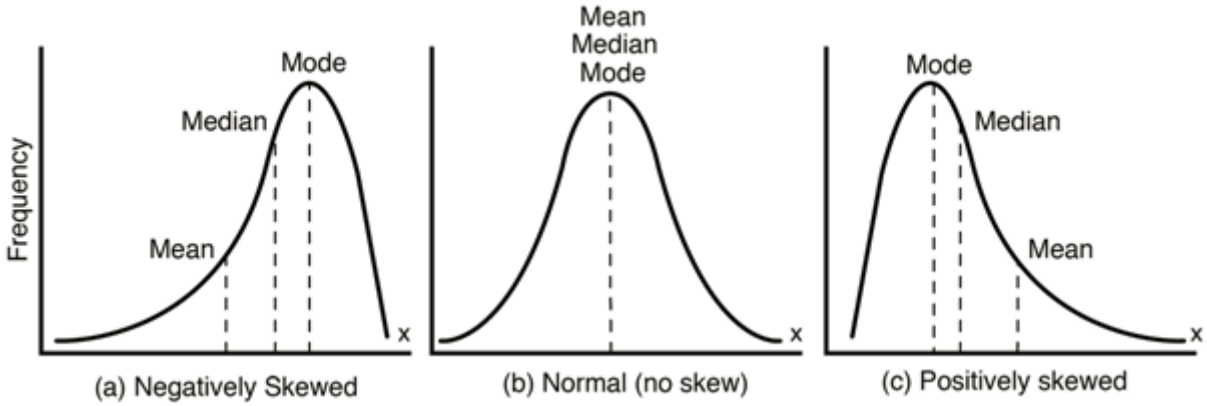
If mean=median=mode, then the graph is a normal distribution (a symmetrical graph)

A normal distribution, or bell curve, is modeled like this, with the probability of the data landing on the x-axis shown on the y-axis





If we have mode>median>mean or mean>median>mode, then we have a skewed distribution graph:



1e. Skewness

It is a way to check the symmetry of a distribution. A skewness of 0 means it's a normal distribution (mean=median=mode), a positive skewness (right skewed) has a positive asymmetry (mean>median>mode), and a negative skewness (left skewed) has a negative asymmetry (mean<median<mode).

It can be measured by:

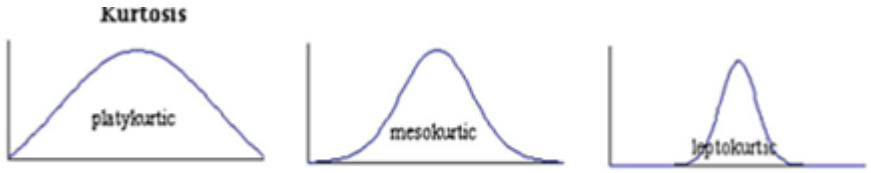
$$Sk = \frac{\overline{x} - mode}{\sigma}$$

Or, it can be found by doing this:

In [ ]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('filename')
df['Price'].skew() # Price is the variable we'll be using in all of our dem
```

1f. Kurtosis

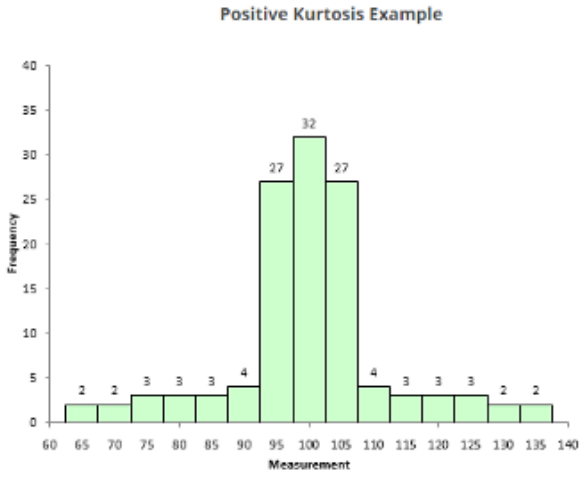
Kurtosis is the measure of the weight of the tails of the distribution (relative to the center of the distribution). A normal distribution has a kurtosis of 3. Also, a distribution that's closely packed together has a higher kurtosis, and a distribution that is spread out has a lower one.

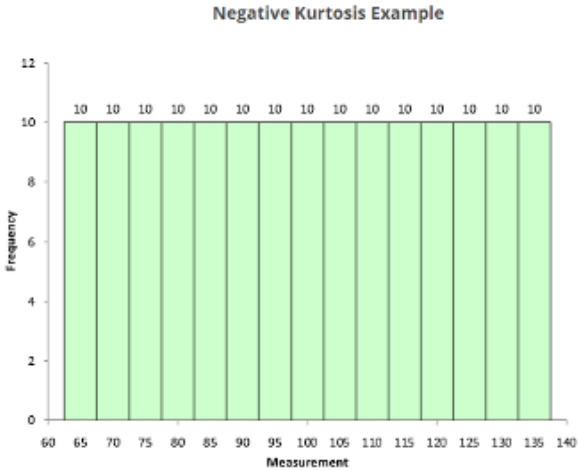


When the distribution of the data is similar to a normal distribution, or the kurtosis of the distribution is 3 (K=3), it is called a *Mesokurtic* distribution.

Any distribution which has a kurtosis more than normal distribution (K>3) is called a *leptokurtic* (thin) distribution. This type of distribution has a positive kurtosis.

Distributions which have a kurtosis that is less than normal distribution (K<3) is called a *platykurtic* (flat) distribution. This type of distribution usually has a negative kurtosis.





Kurtosis can be found like this:

$$Kurtosis\ or\ \kappa = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{4^{th}\ central\ moment}{Variance^2}$$

Or like this (in Python):

In [ ]:
```
df['Price'].kurt()
```

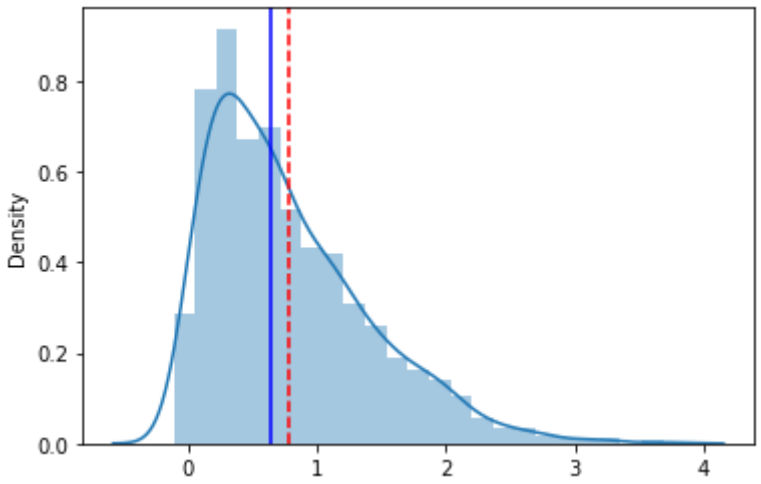
1g. Histograms

We typically use a histogram to graph and observe the central tendencies (mean, median, mode, skewness, etc.), and it can be graphed in matplotlib like so:

In [1]:
```
import numpy as np
import seaborn as sns
from scipy.stats import skewnorm
import matplotlib.pyplot as plt
a = 20 # this is the set value for skewness
r = skewnorm.rvs(a, size=1000)
mean= np.mean(r)
median = np.median(r)
sns.distplot(r)
plt.axvline(x=median, color = 'blue')
plt.axvline(x=mean, color = "red", linestyle='--')
# the blue line is the median, and the red is the mean
```

```
/Users/jadenchen/opt/anaconda3/lib/python3.8/site-packages/seaborn/distri
butions.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot
` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[1]:
```
<matplotlib.lines.Line2D at 0x7fd8c09f62e0>
```



From above, we can see that the box plot is positively skewed, since mean<median

2. Measures of Dispersion

These data measures focus on how much the data varies in a population.

Dispersion provides a wide range of information about the data and is used as a *key concept* to *detect and handle outliers*. It also provides information about the data distribution and answers key questions about the *trend of a variable*.

2a. Range

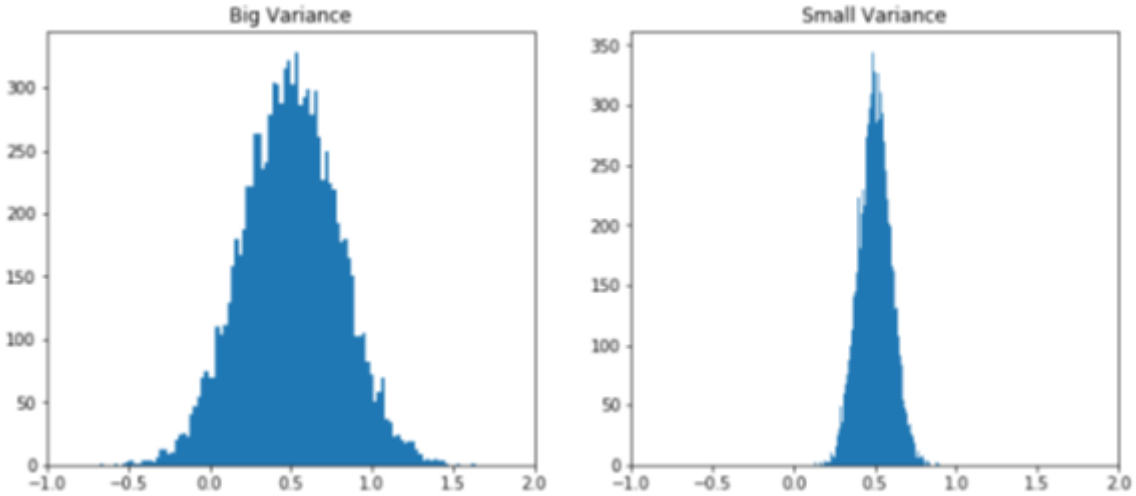
Range is the difference between the maximum and minimum values, found by:

In [ ]:
```
range_of_set=x.max()-x.min()
```

2b. Standard deviation and Variance

Standard deviation is a measure of the variation or dispersion of a set of values. Standard deviations are calculated as the square root of variance. They are greatly affected by outliers as they impact the mean, which changes the standard deviation.

Variance - a measure of how far the data points are from the mean. A low standard deviation/variance indicates that the values are closer to the mean, whereas a high standard deviation/variance is an indication of extreme values or skewness of the data. Standard deviation, like dispersion, is used as a metric to analyze the distribution of data.



Variance calculation:

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

It can also be found like this:

In [ ]:
```
df['Price'].var()
# the standard deviation is found below:
sqrt(df['Price'].var())
```

2c. Interquartile Range

Quantiles are cut points dividing the range of a probability distribution into continuous intervals with equal probabilities. The 1st quantile is the bottom 25% of data, the 2nd is the 25th to 50th percentile of data, the 3rd is the 50th to 75th percentile of data, and the 4th quantile is the top 25% of data. Q1 is the precise 25th percentile, Q2 is the median (the precise 50th percentile), and Q3 is the precise 75th percentile.

Interquartile range is: IQR=Q3-Q1 (50% of the data lies here between Q3 and Q1)

Quantiles can be found like so:

In [ ]:
```
print(df['Price'].quantile(0.75)) # the 0.75 means the 75th percentile
# it can be modified to any percentile between 0 and 1
```

2d. Outliers

Sometimes there are very extreme data points, and these are called outliers. Most of the time, they aren't calculated in the standard deviation, variance, and sometimes not even in the range.

Anything below Q1-1.5x or anything above Q3+1.5x is an outlier (where x=interquartile range)

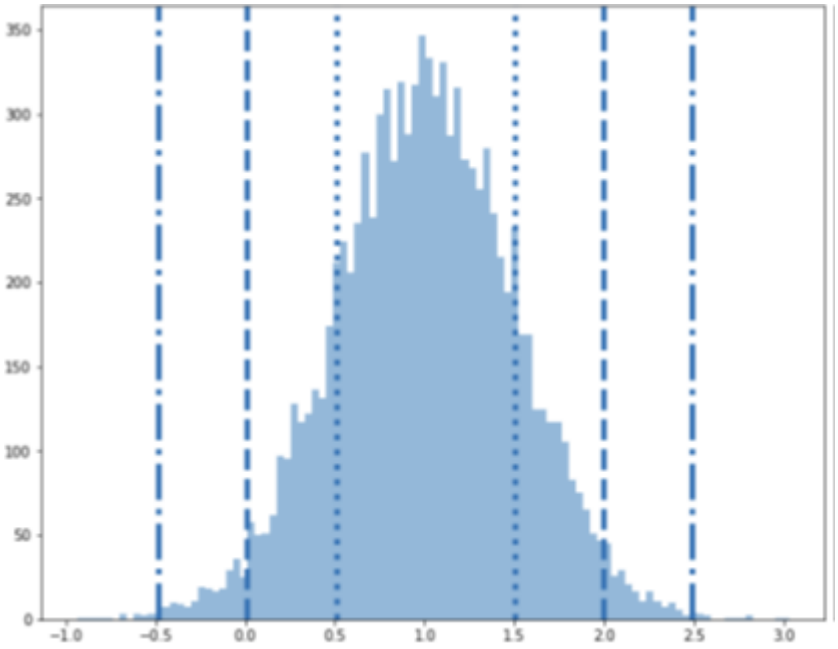
2e. Standard Score

The standard score, or z-score, is another measure for detecting outliers. It takes a data point and uses the number of standard deviations the entry has from the mean as a measure. It takes the population variance into consideration, as so:

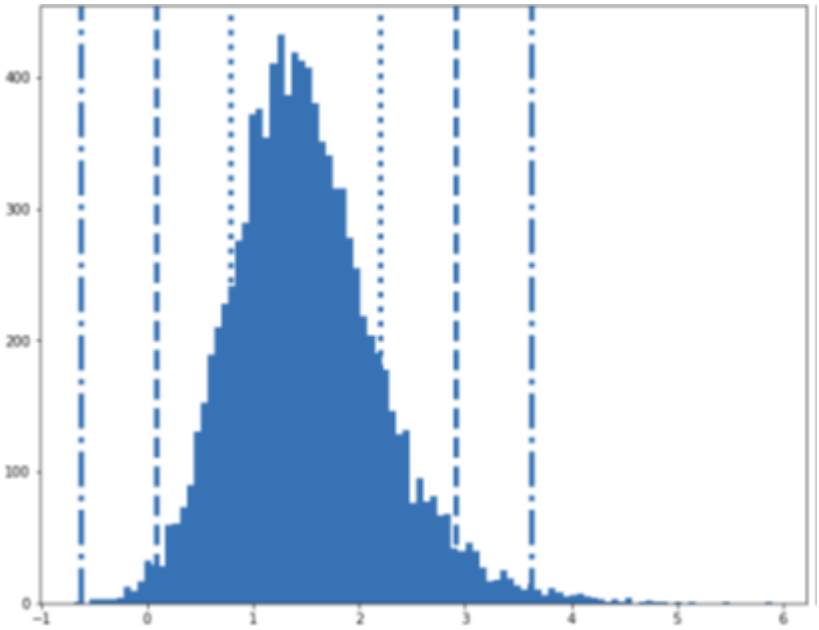
$$z = \frac{x - \overline{x}}{\sigma}$$

If the underlying distribution is normal, a z-score that is greater than 3 or less than 0 only has a probability of around 0.27% of showing up. Furthermore, Chebyshev's theorem states that at most 1/(k^2), where k is an integer, of the total population can fall outside k standard deviations.

Below in the normal distribution, the dotted line indicates the location where z = ±1. The dashed line indicates the location of z = ±2. The dashed-dotted line indicates the location of z = ±3:



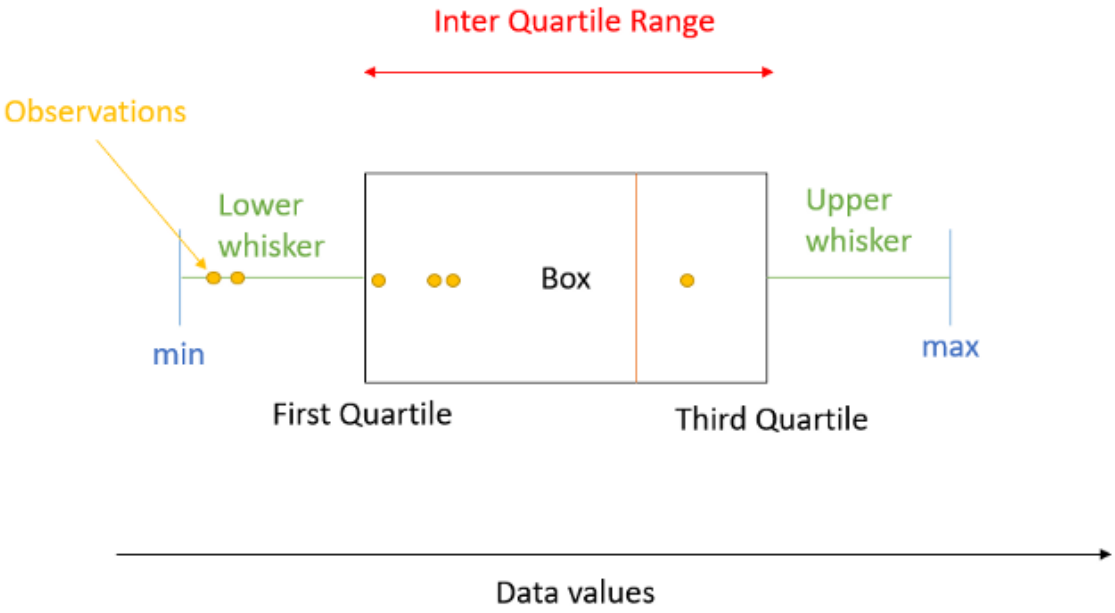
In a skewed distribution, the z values are as shown:



(Note: A downside of the z-score is that the mean itself can also be influenced by extreme outliers. In this case, the median can replace the mean to remove this effect.)

2f. Box Plots

Dispersion measures are commonly graphed on box plots, which generally look like this:

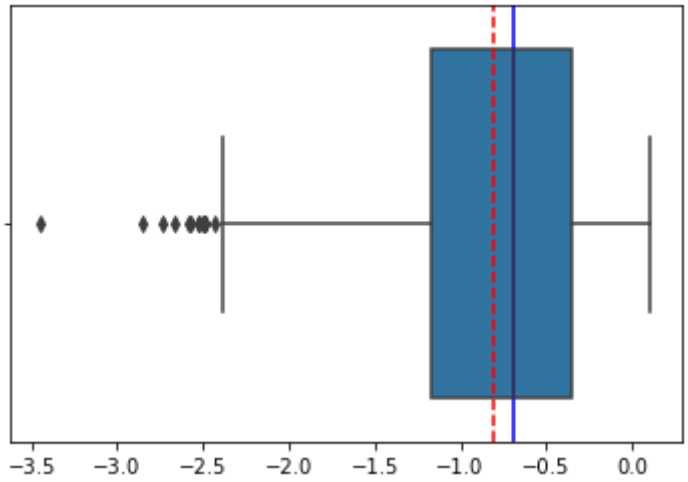


To graph a box plot in matplotlib, do this:

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
a = -14 # this is the set value for skewness
r = skewnorm.rvs(a, size=1000)
mean= np.mean(r)
median = np.median(r)
sns.boxplot(r, orient='h')
plt.axvline(x=median, color = 'blue')
plt.axvline(x=mean, color = "red", linestyle='--')
# the blue line is the median, and the red is the mean
```

```
/Users/jadenchen/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decor
ators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

Out[9]:
```
<matplotlib.lines.Line2D at 0x7fd743260130>
```



From above, we can see the outliers as the black dots, and also see Q1, Q3, and the central tendencies

We can also use the Plotly data library to enhance data visualization, as so:

In [ ]:
```
!pip install plotly
!pip install cufflinks
import plotly.express as px
df = px.data.tips()
df.head()
fig = px.box(df, y="total_bill")
fig.show()
```

This code would give a graph of a box plot, as shown:

3. Bivariance and Multivariate Statistics

These are ways we can analyze the relationship between multiple variables.

3a. Correlation

Correlation is the degree of association between two variables. Remember that CORRELATION IS NOT CAUSATION. A correlation is on a scale from -1 to 1, with -1 and 1 being very similar trends, and 0 being no trend/relation between two variables.

In a -1 or 1 correlation data set, the line or equation of best fit fits all the data points one to one, showing a very very strong correlation between the two variables.

The graph below shows the calculation of correlation and the line of best fit for a data set:

X'	Y'
18.3333	24.06666667

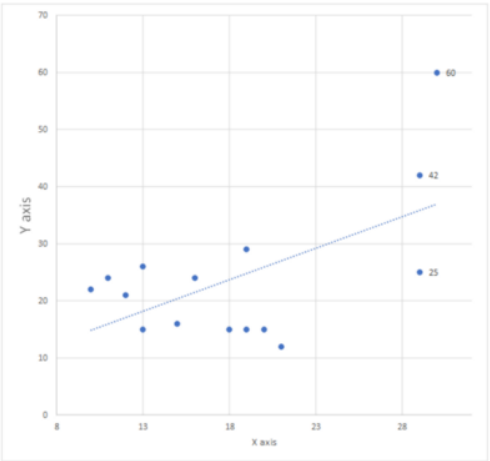
Correlation is always between -1 and +1

		A	B			
X	Y	X-X'	Y-Y'	(X-X') ^2	(Y-Y') ^2	A*B
10	22	-8.3	-2.1	69.4	4.3	17.2
11	24	-7.3	-0.1	53.8	0.0	0.5
12	21	-6.3	-3.1	40.1	9.4	19.4
13	26	-5.3	1.9	28.4	3.7	-10.3
13	15	-5.3	-9.1	28.4	82.2	48.4
15	16	-3.3	-8.1	11.1	65.1	26.9
16	24	-2.3	-0.1	5.4	0.0	0.2
18	15	-0.3	-9.1	0.1	82.2	3.0
19	15	0.7	-9.1	0.4	82.2	-6.0
19	29	0.7	4.9	0.4	24.3	3.3
20	15	1.7	-9.1	2.8	82.2	-15.1
21	12	2.7	-12.1	7.1	145.6	-32.2
29	25	10.7	0.9	113.8	0.9	10.0
29	42	10.7	17.9	113.8	321.6	191.3
30	60	11.7	35.9	136.1	1,291.2	419.2
Sum				611.3	2,194.9	675.7
Sx					6.6	
Sy						12.5
Pearson Correlation coeeficient						0.583

$$r = \frac{1}{n-1}\left(\frac{\sum_x \sum_y (x-\bar{x})(y-\bar{y})}{s_x s_y}\right)$$

$$s_x = \sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$$

X' - Mean of X
Y' - Mean of Y
n - Total data points



Mean	24.07
standard deviation	12.52
Mean - 3*std	-0.98
Mean - 3*std	49.11

Quartile 1	15
Quartile 3	25.5
IQR	10.5
Q1-1.5*IQR	-0.75
Q3+1.5*IQR	41.25

Change it to 2.5*std and check

3b. Covariance

The word covariance is often incorrectly used as correlation. There are a number of fundamental differences between these two. Covariance usually measures the extent to which the two variables are dependent on each other, while correlation focuses more on the strength of variability of these two variables.

It's value lies in all real numbers, while correlation lies in between -1 and 1.

3c. Cross-tabulation

Cross-tabulation can be treated as an "eyeball" version of correlation detection for categorical variables. It helps pinpoint innumerable insights in a data set and find information through discrete detection.

An example: This is a list of weather information and another list of a golfer's decisions on whether or not he will go golfing. The crosstab() function generates the following table:

In [6]:
```
import pandas as pd
weather = ["rainy","sunny","rainy","windy","windy",
           "sunny","rainy","windy","sunny","rainy",
           "sunny","windy","windy"]
golfing = ["Yes","Yes","No","No","Yes","Yes","No","No",
           "Yes","No","Yes","No","No"]
dfGolf = pd.DataFrame({"weather":weather,"golfing":golfing})
pd.crosstab(dfGolf.weather, dfGolf.golfing, margins=True)
```

Out[6]:

golfing	No	Yes	All
weather			
rainy	3	1	4
sunny	0	4	4
windy	4	1	5
All	7	6	13

From above, we see that the columns and rows give all the data inputted. For a dataset with a limited number of features, this is a handy way to inspect imbalance or bias (relating to the golfer). We can tell that the golfer goes golfing if the weather is sunny, and that they seldom go golfing on rainy or windy days. This is an example of the method of cross-tabulation.

4. Categorial Variables

When you read a file using Pandas, it will automatically assign any variable that is a number as a numerical variable. Howevever, there are more than one type of variable.

Categorial variables are variables that are not relating to numbers (even if they might include numbers), such as a zip code, date, or time, unlike numerical values, which are related to size, numerica values, and can be modified by operators.

Categorial distribution don't have a mean or median, but they have a mode.

It is up to the data scientist to convert between these two types of variables if needed, since Pandas isn't perfect.

We can use Counter to graph and interpret categorial variables.