Tutorial #1 (part 1)

Fundamentals of Analytics (Descriptive Statistics)

Objective

The objective of this tutorial is to:

- familiarize participants with the tools involved in analytics projects
- be able to calculate simple descriptive statistics and plot simple charts
- check hypotheses about the mean (one sample and two sample)

This tutorial will cover the following topics:

1. Descriptive statistics
   A. Measures of location and variation
   B. Bivariate relationships
2. Simple charting and visualization
3. Hypothesis testing

Tools: Jupyter notebooks, Python with the following libraries: numpy, pandas, matplotlib, scipy, statsmodels.

Prerequisites: Basic Python knowledge and familiarity with descriptive statistics.

Descriptive Statistics

Import the requried libraries.

In [1]:

```
from scipy.stats import *
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Lets create a Numpy array with some sample values.

In [2]:

```
d = np.array([5, 8, 55, 8, 7, 6, 5, 4, 5, 9, 11])
```

Lets find out the frequency of the items in the array.

In [3]:

```
# Frequency of items in the array
stats.itemfreq(d)
```

```
c:\python37\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning:
`itemfreq` is deprecated!
`itemfreq` is deprecated and will be removed in a future version. Use inst
ead `np.unique(..., return_counts=True)`
```

Out[3]:

```
array([[ 4,  1],
       [ 5,  3],
       [ 6,  1],
       [ 7,  1],
       [ 8,  2],
       [ 9,  1],
       [11,  1],
       [55,  1]], dtype=int64)
```

Since *itemfreq* is going to be deprecated, we can use *np.unique(..., return_counts=True)* instead

In [4]:

```
np.unique(d, return_counts=True)
```

Out[4]:

```
(array([ 4,  5,  6,  7,  8,  9, 11, 55]),
 array([1, 3, 1, 1, 2, 1, 1, 1], dtype=int64))
```

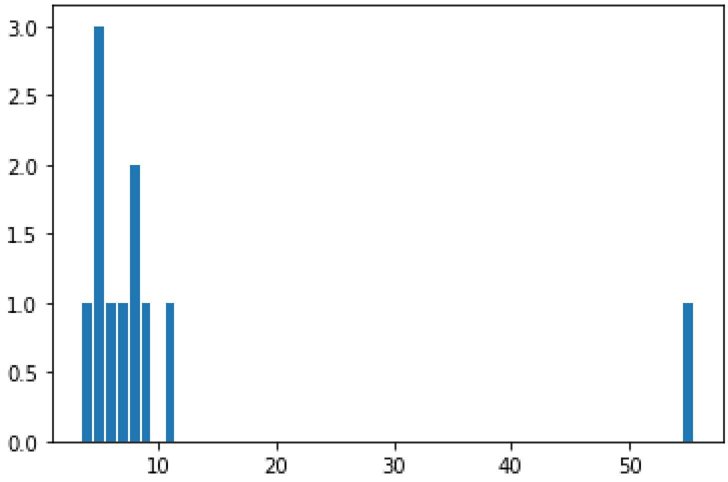
A bar chart can be used to visualize the frequency of the values.

In [5]:

```
values, frequency = np.unique(d, return_counts=True)
plt.bar(values, height=frequency)
```

Out[5]:

```
<BarContainer object of 8 artists>
```



What is the **mode** of this data?

In [6]:

```
# Find out the mode
stats.mode(d)
```

Out[6]:

```
ModeResult(mode=array([5]), count=array([3]))
```

Alternatively, we can do this in **Pandas**.

Let's first convert our numpy array to a Pandas **dataframe**. We can name the column whatever we want. We will call it 'd'.

In [7]:

```
df = pd.DataFrame(d)
df.columns = ['d']
```

In [8]:

```
df.mode() # For the whole dataframe
```

Out[8]:

	d
0	5

We can similarly find the mean and the median.

In [9]:

```
print("The mean is", df['d'].mean())
print("The median is", df['d'].median())
```

```
The mean is 11.181818181818182
The median is 7.0
```

The *quantile()* function gives the percentiles. The percentile required is the parameter passed to the quantile function.

In [10]:

```
df['d'].quantile(0.5)
```

Out[10]:

```
7.0
```

We can use the *std()* and *var()* functions to find out the standard deviation and variance. Verify that squaring the standard deviation gives the variance.

In [11]:

```
print("Standard deviation =",df['d'].std())
print("Variance =", df['d'].var())
```

```
Standard deviation = 14.682085559062664
Variance = 215.5636363636364
```

Find the coefficient of variation.

In [12]:

```
df['d'].std()/df['d'].mean()
```

Out[12]:

```
1.3130320418673926
```

Descriptive statistics for a dataframe can be obtained using the *describe()* method.

In [13]:

```
df.describe()
```

Out[13]:

	d
count	11.000000
mean	11.181818
std	14.682086
min	4.000000
25%	5.000000
50%	7.000000
75%	8.500000
max	55.000000

Analysis of cricket data

Let's read a data set for analysis.

In [14]:

```
# to read from a locally saved file use this
# df = pd.read_csv("men_odi_india.csv")

# to read directly from githib use the following path
df = pd.read_csv("https://raw.githubusercontent.com/agrianalytics/fundamentals/master/men_odi_india.csv")
```

In [15]:

```
df.dtypes
```

Out[15]:

```
Date              object
Player            object
Runs             float64
NotOut              bool
Minutes          float64
BallsFaced       float64
Fours            float64
Sixes            float64
StrikeRate       float64
Innings          float64
Participation     object
Opposition        object
Ground            object
dtype: object
```

In [16]:

```
df.shape
```

Out[16]:

```
(10642, 13)
```

In [17]:

```
df.head()
```

Out[17]:

	Date	Player	Runs	NotOut	Minutes	BallsFaced	Fours	Sixes	StrikeRate	Innings
0	13-11-2014	RG Sharma	264.0	False	225.0	173.0	33.0	9.0	152.601156	1.0
1	08-12-2011	V Sehwag	219.0	False	208.0	149.0	25.0	7.0	146.979866	1.0
2	02-11-2013	RG Sharma	209.0	False	222.0	158.0	12.0	16.0	132.278481	1.0
3	13-12-2017	RG Sharma	208.0	True	212.0	153.0	13.0	12.0	135.947712	1.0
4	24-02-2010	SR Tendulkar	200.0	True	226.0	147.0	25.0	3.0	136.054422	1.0

Which player has the highest average?

In [18]:

```
df.groupby(['Player'])['Runs'].mean().sort_values(ascending=False)[0:1]
```

Out[18]:

```
Player
GK Khoda    57.5
Name: Runs, dtype: float64
```

Plot a histogram of runs scored. What does it tell us?

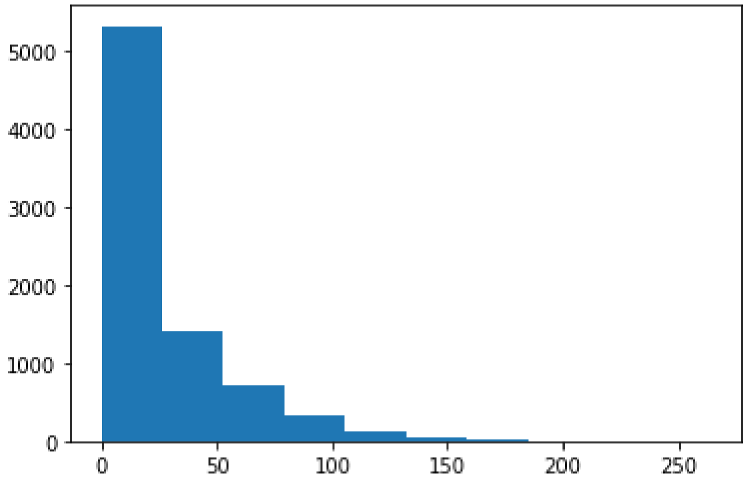
In [19]:

```
plt.hist(df['Runs'])
```

```
c:\python37\lib\site-packages\numpy\lib\histograms.py:824: RuntimeWarning:
invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
c:\python37\lib\site-packages\numpy\lib\histograms.py:825: RuntimeWarning:
invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

Out[19]:

```
(array([5.320e+03, 1.419e+03, 7.190e+02, 3.170e+02, 1.330e+02, 5.000e+01,
        1.100e+01, 4.000e+00, 1.000e+00, 1.000e+00]),
 array([  0. ,  26.4,  52.8,  79.2, 105.6, 132. , 158.4, 184.8, 211.2,
        237.6, 264. ]),
 <a list of 10 Patch objects>)
```



How are *Runs, Minutes, BallsFaced, Fours, Sixes and StrikeRate* related?

In [20]:

```
df[['Runs', 'Minutes', 'BallsFaced', 'Fours', 'Sixes', 'StrikeRate']].corr()
```

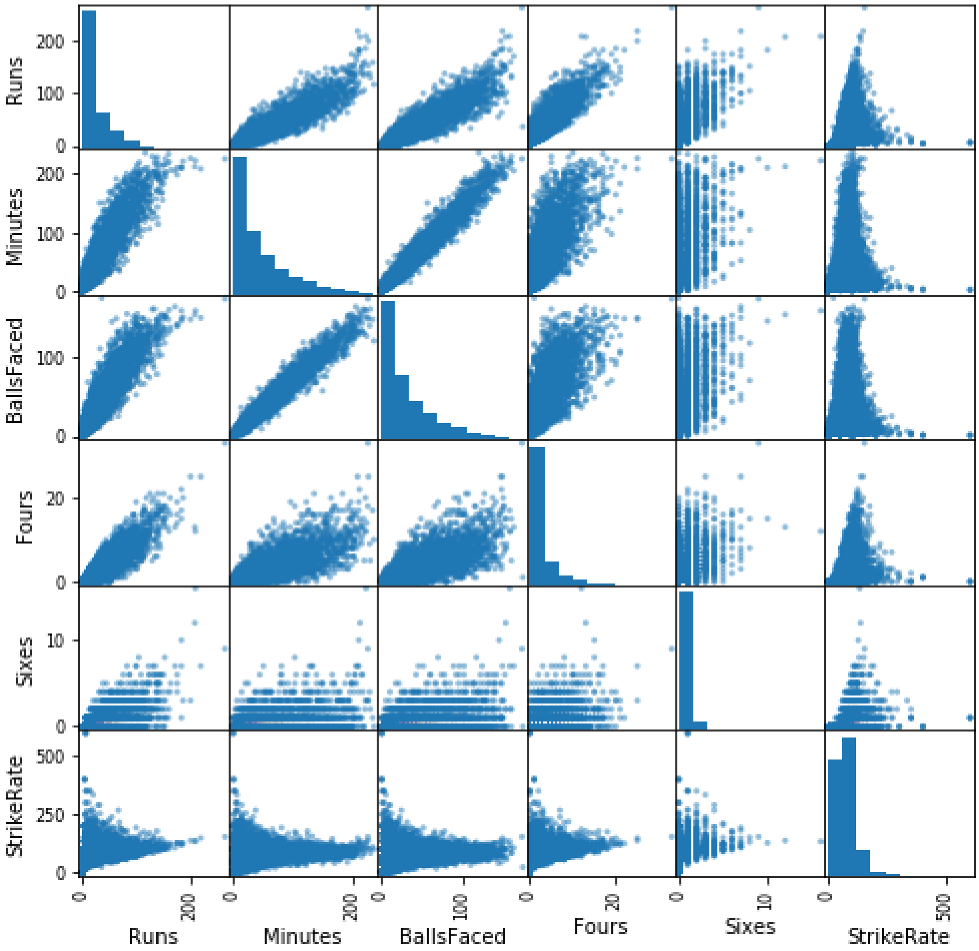
Out[20]:

	Runs	Minutes	BallsFaced	Fours	Sixes	StrikeRate
Runs	1.000000	0.920129	0.921713	0.892633	0.597354	0.376375
Minutes	0.920129	1.000000	0.978482	0.784559	0.428620	0.195901
BallsFaced	0.921713	0.978482	1.000000	0.776376	0.431885	0.173756
Fours	0.892633	0.784559	0.776376	1.000000	0.439221	0.385614
Sixes	0.597354	0.428620	0.431885	0.439221	1.000000	0.354869
StrikeRate	0.376375	0.195901	0.173756	0.385614	0.354869	1.000000

We can visualize this relationship using scatter plots.

In [21]:

```
fig = pd.plotting.scatter_matrix(df[['Runs', 'Minutes', 'BallsFaced',
                                      'Fours', 'Sixes', 'StrikeRate']],
                                 figsize = (8, 8))
```



Can we compare the performance of 'SR Tendulkar' and 'V Sehwag'?

Find out the:

- mean
- standard deviation and
- *standard error* of the mean

of the runs scored by these two players: 'SR Tendulkar' and 'V Sehwag'.

In [22]:

```
df[df.Player.isin(['SR Tendulkar', 'V Sehwag'])].groupby(['Player'])['Player', 'Runs'].
mean()
```

Out[22]:

```
                  Runs
       Player
 SR Tendulkar  40.765487
     V Sehwag  34.021277
```

In [23]:

```
df[df.Player.isin(['SR Tendulkar', 'V Sehwag'])].groupby(['Player'])['Player','Runs'].s
td()
```

Out[23]:

```
                  Runs
       Player
 SR Tendulkar  40.039480
     V Sehwag  35.351525
```

In [24]:

```
df[df.Player.isin(['SR Tendulkar', 'V Sehwag'])].groupby(['Player'])['Player','Runs'].s
em()
```

Out[24]:

```
               Player      Runs
       Player
 SR Tendulkar     NaN  1.883299
     V Sehwag     NaN  2.306079
```

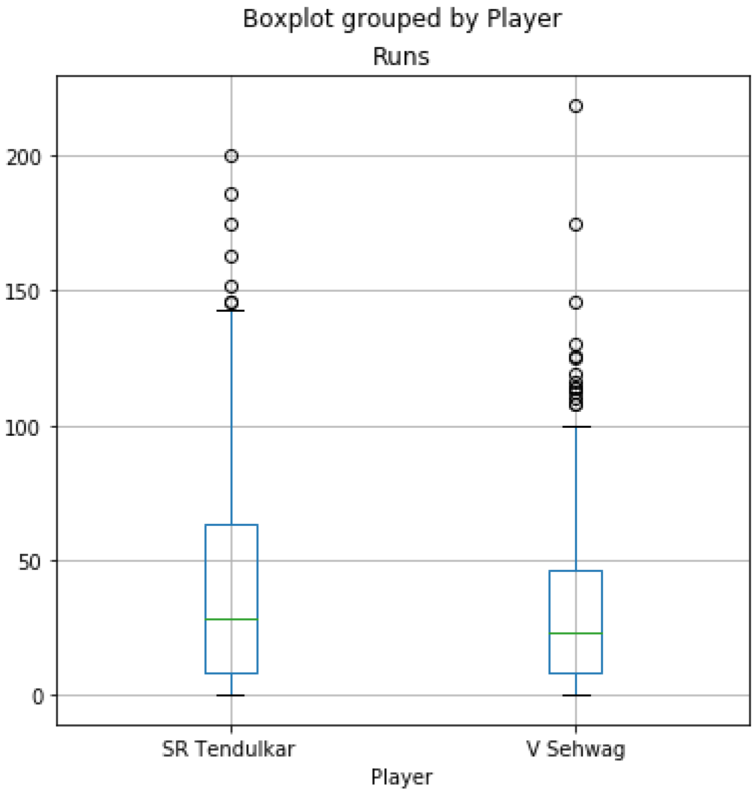
We can also visualize this graphically in the form of boxplots.

In [25]:

```
player_list = ['SR Tendulkar', 'V Sehwag']
df[df.Player.isin(player_list)].boxplot(column='Runs', by='Player', figsize=(6, 6))
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x25c490bb7b8>
```



What have we concluded? Does the performance of these two players differ?

Hypothesis Testing

The mean score of 'SR Tendulkar' is 42. Do you support this hypothesis? Write down the null and alternative hypotheses and check its validity.

What can you say? What about 41, 42, ...?

At what confidence level is your conculsion valid?

In [26]:

```
ten_runs = df[df.Player == 'SR Tendulkar']['Runs']
ten_runs.dropna(inplace=True)
stats.ttest_1samp(ten_runs, 40)
```

```
c:\python37\lib\site-packages\pandas\core\series.py:4303: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(result)
```

Out[26]:

```
Ttest_1sampResult(statistic=0.406460598899358, pvalue=0.6845970044555593)
```

Is there a *statistically* significant difference in the average score of 'SR Tendulkar' and 'V Sehwag'?

In [27]:

```
seh_runs = df[df.Player == 'V Sehwag']['Runs']
seh_runs.dropna(inplace=True)
stats.ttest_ind(ten_runs, seh_runs)
```

Out[27]:

```
Ttest_indResult(statistic=2.1780573622467294, pvalue=0.029742142199471473)
```