Distribution

- A distribution of data is a representation (or function) showing all possible values (or intervals) and how often those values occur.
- For **categorical data**, we'll often see percentage or exact number for each of the category.
- For **numerical data**, we'll see the data split into appropriate sized buckets ordered from smallest to largest
- When a distribution is plotted into a graph, we can observe different shapes of the curve. Based on the shape and other attributes, there exists many types of distributions. A few statistical distributions are,
    - Bernoulli Distribution
    - Binomial Distribution
    - Cumulative frequency distribution
    - Bimodal distribution
    - Gaussian distribution (Normal distribution)
    - Uniform distribution

```
import math
import numpy as np
import pandas as pd
from matplotlib import pyplot
from scipy import stats

matches    = pd.read_csv('../input/matches.csv')
deliveries = pd.read_csv('../input/deliveries.csv')
```

Cumulative relative frequency graph

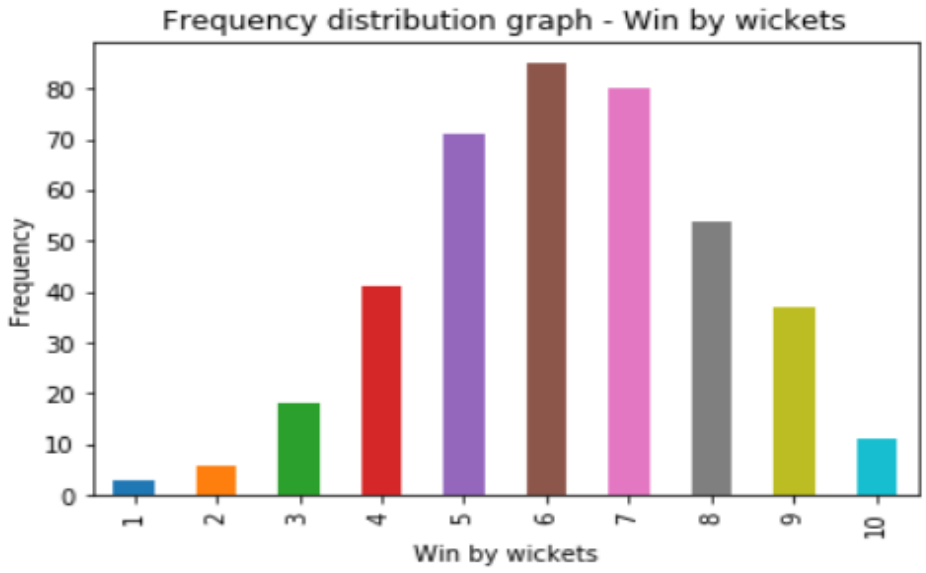
Let's take win_by_wickets dataset and plot a frequency distribution graph.

X-axis - Win by wickets (value from 1 to 10), Y-axis - Number of instances (or frequency) of win-by-wicket margin

```
win_by_wickets_data = matches[matches.win_by_wickets > 0].win_by_wic
kets
win_by_wickets_freq = win_by_wickets_data.value_counts(sort=False)
print(win_by_wickets_freq)
plt = win_by_wickets_freq.plot.bar()
plt.set_title("Frequency distribution graph - Win by wickets")
plt.set_xlabel("Win by wickets")
plt.set_ylabel("Frequency")
```

```
1      3
2      6
3     18
4     41
5     71
6     85
7     80
8     54
9     37
10    11
Name: win_by_wickets, dtype: int64

Text(0, 0.5, 'Frequency')
```

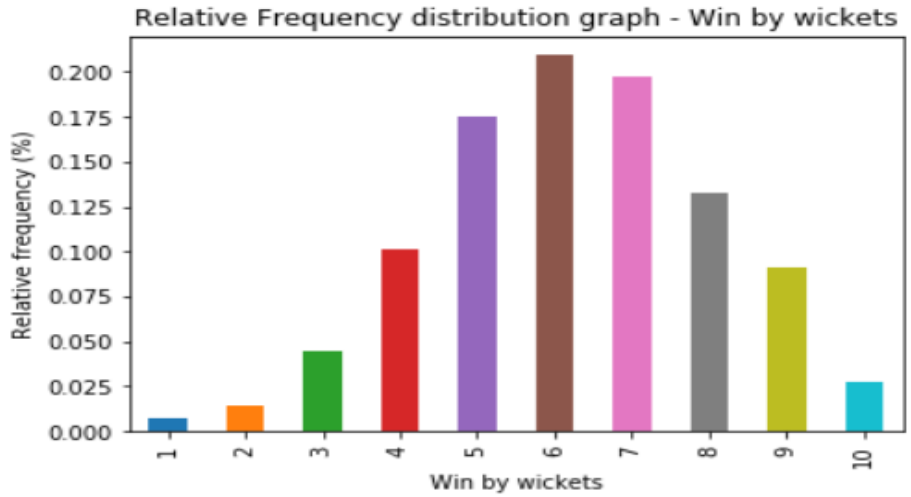


Relative frequency distribution graph for the same data. Here in **Y-axis**, instead of showing the frequency, we show the **percentage** of the value. We can use normalize = True argument for pandas.Series.value_counts method

```
win_by_wickets_rel_freq = win_by_wickets_data.value_counts(sort = False, normalize = True)
print(win_by_wickets_rel_freq)
plt = win_by_wickets_rel_freq.plot.bar()
plt.set_title("Relative Frequency distribution graph - Win by wickets")
plt.set_xlabel("Win by wickets")
plt.set_ylabel("Relative frequency (%)")
```

```
1     0.007389
2     0.014778
3     0.044335
4     0.100985
5     0.174877
6     0.209360
7     0.197044
8     0.133005
9     0.091133
10    0.027094
Name: win_by_wickets, dtype: float64

Text(0, 0.5, 'Relative frequency (%)')
```

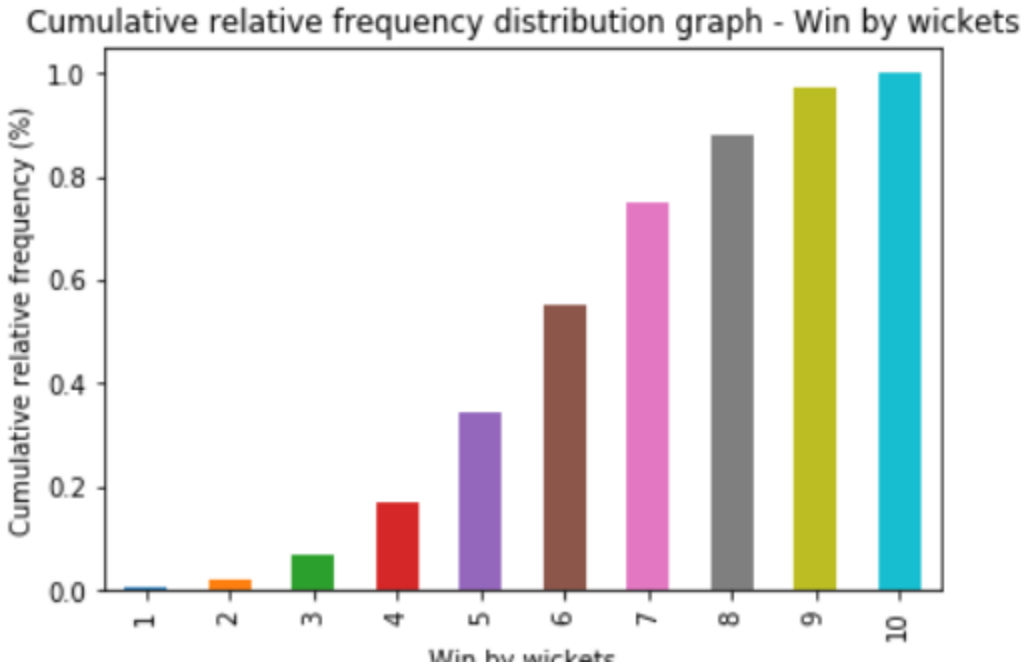


cumulative relative frequency graph using `pandas.Series.cumsum`

```
win_by_wickets_cumulative_freq = win_by_wickets_data.value_counts(so
rt = False, normalize = True).cumsum()
print(win_by_wickets_cumulative_freq)
plt = win_by_wickets_cumulative_freq.plot.bar()
plt.set_title("Cumulative relative frequency distribution graph - Wi
n by wickets")
plt.set_xlabel("Win by wickets")
plt.set_ylabel("Cumulative relative frequency (%)")
```

```
1     0.007389
2     0.022167
3     0.066502
4     0.167488
5     0.342365
6     0.551724
7     0.748768
8     0.881773
9     0.972906
10    1.000000
Name: win_by_wickets, dtype: float64


Text(0, 0.5, 'Cumulative relative frequency (%)')
```



Normal distribution

Normal distribution is a continuous probability distribution that describes many natural datasets. It is also known as **bell curve** or **Gaussian distribution**. We see many natural examples that are closer to a normal distribution.

- Heights of people
- Shoe sizes
- Lap duration in a car race

In a perfect normal distribution, we can see 50% symmetry about the center. Also, the center is - **mean** = **mode** = **median**.

Normal distribution and variance

If the variance of the dataset is high, the curve tends to look flat. If variance is low, curve is more steeper.

Let's plot win_by_wickets data and watch the curve.

```
# Get mean (mu) and std (sigma)
win_by_wickets_mean, win_by_wickets_std = win_by_wickets_data.me
an(), win_by_wickets_data.std()

# Plot histogram (normalized) - LIGHT-BLUE
win_by_wickets_data.hist(color='lightblue', weights = np.zeros_l
ike(win_by_wickets_data) + 1.0 / win_by_wickets_data.count())

# Plot line graph - RED
win_by_wickets_data.value_counts(sort=False, normalize=True).plo
t.line(color='red')

# Normal distribution for random points between 1 to 10 with mean,
std.
random_data = np.arange(1, 10, 0.001)
pyplot.plot(random_data, stats.norm.pdf(random_data, win_by_wick
ets_mean, win_by_wickets_std), color='green')
```