# 1 Continuous Distributions : ¶

In [ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import Math, Latex
from IPython.core.display import Image
import numpy as np
import seaborn as sns
```

In [ ]:

```
#setting plotting style
sns.set(color_codes=True)
#Setting plotting style
sns.set(rc={"figure.figsize":(5,5)})
```

## 1.1 Uniform Distributions:

Same results for all inputs

In [ ]:

```
from scipy.stats import uniform
```

In [ ]:

```
n = 10000
start = 10
width = 20
data_uniform = uniform.rvs(size=n, loc=start, scale=width)
```

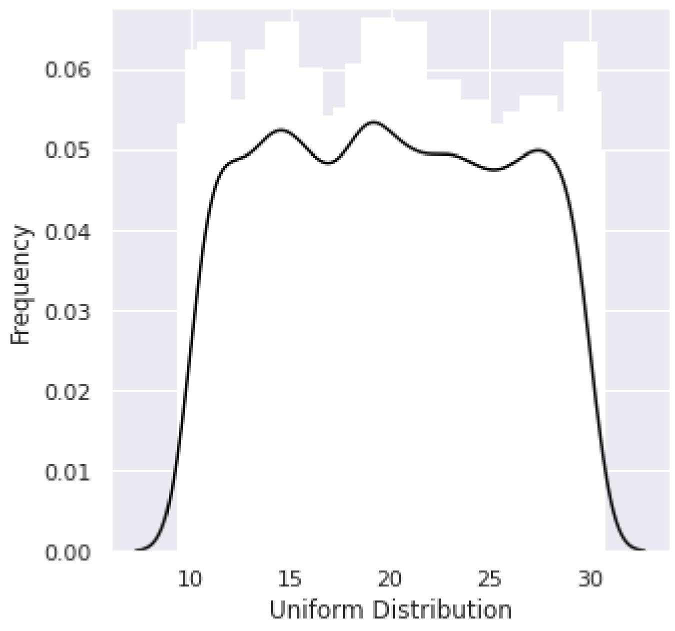
In [ ]:

```
ax = sns.distplot(
    data_uniform,
    bins=100,
    kde=True,
    color="black",
    hist_kws={"linewidth":15,"alpha":1})
ax.set(xlabel='Uniform Distribution', ylabel='Frequency')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]:

```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Uniform Distribution')]
```



## 1.2 Normal Distributions:

In [ ]:

```
from scipy.stats import norm
```

In [ ]:

```
data_norm = norm.rvs(size=10000, loc=0, scale=1)
```

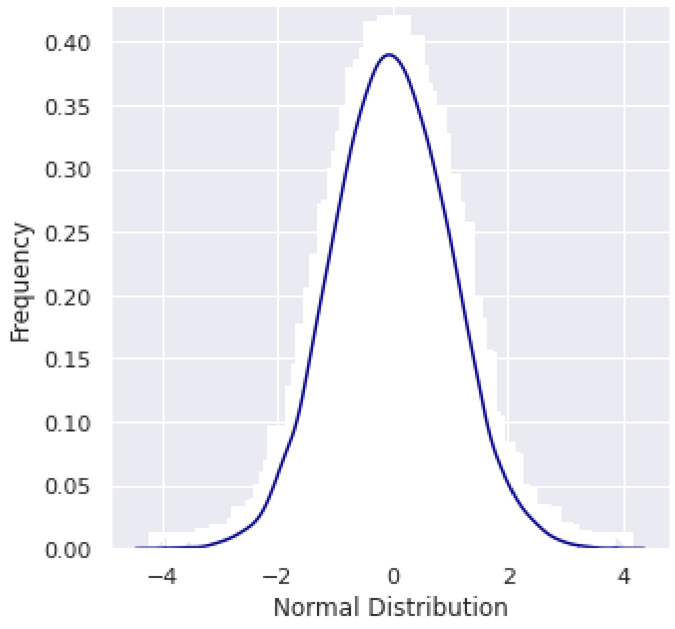
In [ ]:

```
ax = sns.distplot(
    data_norm,
    bins=100,
    kde=True,
    color='darkblue',
    hist_kws={"linewidth":15, "alpha":1}
)
ax.set(xlabel="Normal Distribution", ylabel="Frequency")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:

```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Normal Distribution')]
```



## 1.3 Exponential Distributions:

1. To find out the time required between two discrete values.
2. Time between two patients

In [ ]:

```
from scipy.stats import expon
```

In [ ]:

```
expo_data = expon.rvs(loc=0, scale=1, size=10000)
```

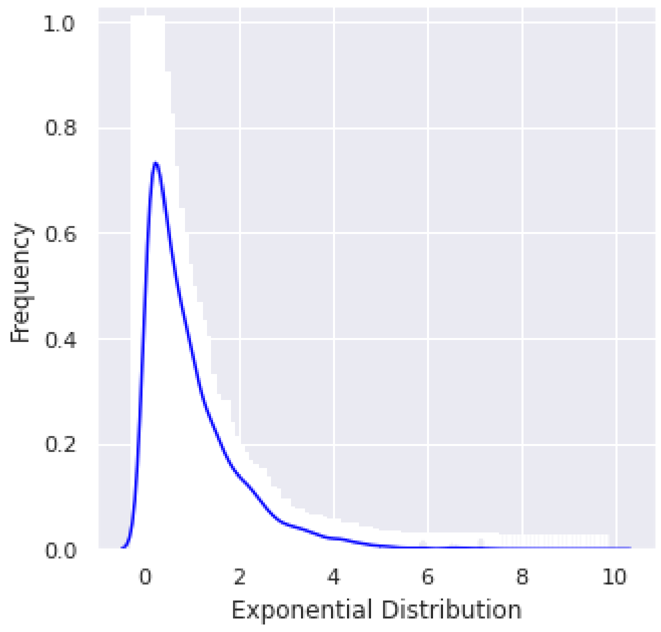
In [ ]:

```
ax = sns.distplot(expo_data,
                  kde=True,
                  bins=100,
                  color="blue",
                  hist_kws={'linewidth':15, 'alpha':1})
ax.set(xlabel='Exponential Distribution', ylabel='Frequency')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]:

```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Exponential Distribution')]
```



## 1.4 Chi-square Distribution:

1. To check whether actual data value is same as expected value.
2. Uses degree of freedom.

In [ ]:

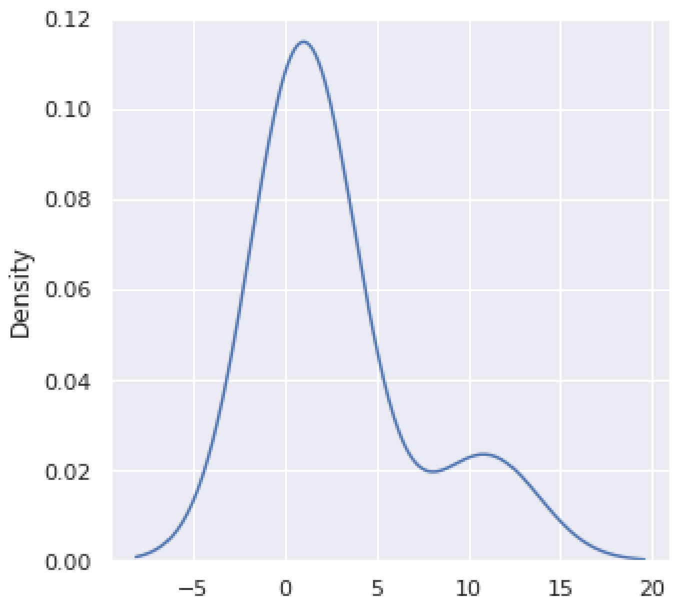
```
from numpy import random
x = random.chisquare(df=2, size=(2,3))
print(x)
```

```
[[ 1.25249363 10.99857441  0.62661097]
 [ 1.03203813  0.48734937  1.6799819 ]]
```

In [ ]:

```
sns.distplot(x, hist=False)
plt.show()
```

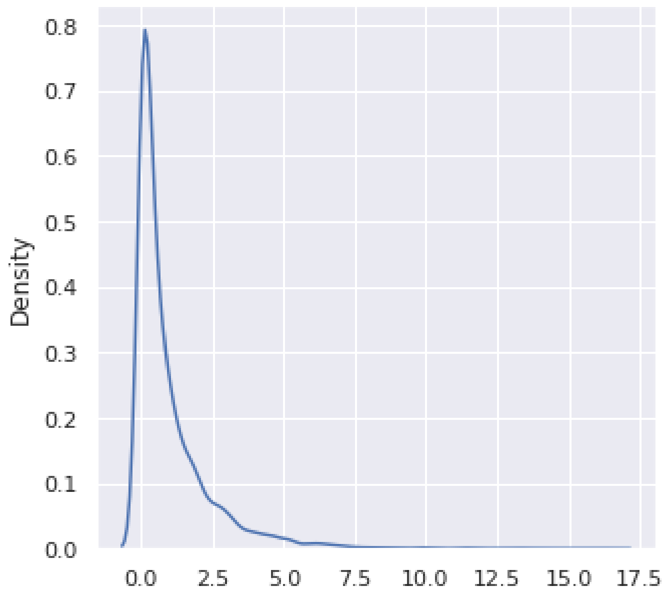
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `kdeplot` (an axes-level function for kern
el density plots).
  warnings.warn(msg, FutureWarning)
```



In [ ]:

```
sns.distplot(random.chisquare(df=1, size=10000), hist=False)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `kdeplot` (an axes-level function for kern
el density plots).
  warnings.warn(msg, FutureWarning)
```



## 1.5 Weibull Distribution:

1. If data doesn't match any other distribution then we can say it follows weibull distribution.

In [ ]:

```
#shape
a = 5
s = np.random.weibull(a,1000)
```

In [ ]:

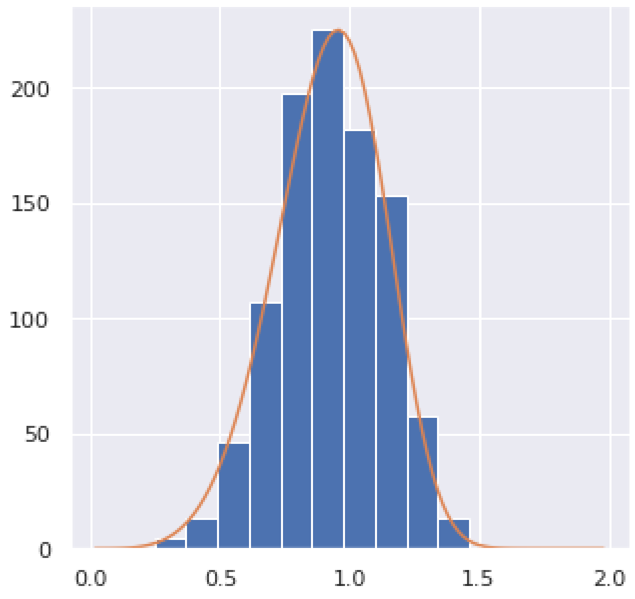
```
x = np.arange(1,100.)/50.

def weib(x, n, a):
  return (a / n) * (x / n)**(a - 1) * np.exp(-(x / n)**a)
```

In [ ]:

```
count, bins, ignored = plt.hist(np.random.weibull(5., 1000))

x = np.arange(1, 100.)/50.
scale = count.max()/weib(x, 1., 5.).max()
plt.plot(x, weib(x, 1., 5.) * scale)
plt.show()
```


200
150
100
50
0
0.0
0.5
1.0
1.5
2.0