**DISCRETE DISTRIBUTIONS**[**¶**](#gjdgxs)

In [1]:

*# for inline plots in jupyter*  
%**matplotlib** inline  
*# import matplotlib*  
**import** **matplotlib.pyplot** **as** **plt**  
*# for latex equations*  
**from** **IPython.display** **import** Math, Latex  
*# for displaying images*  
**from** **IPython.core.display** **import** Image

In [2]:

*# import seaborn*  
**import** **seaborn** **as** **sns**  
*# settings for seaborn plotting style*  
sns.set(color\_codes=**True**)  
*# settings for seaborn plot sizes*  
sns.set(rc={'figure.figsize':(5,5)})

**UNIFORM DISTRIBUTION**[**¶**](#30j0zll)

You can visualize uniform distribution in python with the help of a random number generator acting over an interval of numbers (a,b). You need to import the uniform function from scipy.stats module.

In [3]:

*# import uniform distribution*  
**from** **scipy.stats** **import** uniform

The uniform function generates a uniform continuous variable between the specified interval via its loc and scale arguments. This distribution is constant between loc and loc + scale. The size arguments describe the number of random variates. If you want to maintain reproducibility, include a random\_state argument assigned to a number.

In [10]:

*# random numbers from uniform distribution*  
n = 10000  
start = 10  
width = 20  
data\_uniform = uniform.rvs(size=n, loc = start, scale=width)

You can use Seaborn’s distplot to plot the histogram of the distribution you just created. Seaborn’s distplot takes in multiple arguments to customize the plot. You first create a plot object ax. Here, you can specify the number of bins in the histogram, specify the color of the histogram and specify density plot option with kde and linewidth option with hist\_kws. You can also set labels for x and y axis using the xlabel and ylabel arguments.

In [11]:

ax = sns.distplot(data\_uniform,  
 bins=100,  
 kde=**True**,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Uniform Distribution ', ylabel='Frequency')

Out[11]:

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

**Bernoulli Distribution**[**¶**](#1fob9te)

**image.png**

In [16]:

**from** **scipy.stats** **import** bernoulli  
data\_bern = bernoulli.rvs(size=10000,p=0.6)

In [17]:

ax= sns.distplot(data\_bern,  
 kde=**False**,  
 color="skyblue",  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Bernoulli Distribution', ylabel='Frequency')

Out[17]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Bernoulli Distribution')]

**BINOMINAL DISTRIBUTION**[**¶**](#3znysh7)

**image.png**

In [14]:

**from** **scipy.stats** **import** binom  
data\_binom = binom.rvs(n=10,p=0.8,size=10000)

In [15]:

ax = sns.distplot(data\_binom,  
 kde=**False**,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Binomial Distribution', ylabel='Frequency')

Out[15]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Binomial Distribution')]

**Poisson Distribution**[**¶**](#2et92p0)

Poisson random variable is typically used to model the number of times an event happened in a time interval image.png

In [12]:

**from** **scipy.stats** **import** poisson  
data\_poisson = poisson.rvs(mu=3, size=10000)

You can generate a poisson distributed discrete random variable using scipy.stats module's poisson.rvs() method which takes μ as a shape parameter and is nothing but the λ in the equation. To shift distribution use the loc parameter. size decides the number of random variates in the distribution. If you want to maintain reproducibility, include a random\_state argument assigned to a number.

In [13]:

ax = sns.distplot(data\_poisson,  
 bins=30,  
 kde=**False**,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Poisson Distribution', ylabel='Frequency')

Out[13]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Poisson Distribution')]

In [ ]: