Binomial distribution

Binomial Distribution

- 'n' identical trials.
- Each trial has only two possible outcomes denoted as success or failure.
- Each trial is independent of the previous trials.

Binomial Distribution Formula

$$P(x) = \binom{n}{x} p^x q^{n-x} = \frac{n!}{(n-x)!\,x!} p^x q^{n-x}$$

where
n = the number of trials (or the number being sampled)
x = the number of successes desired
p = probability of getting a success in one trial
$q = 1 - p$ = the probability of getting a failure in one trial

Binomial Distribution in Python using SciPy package

scipy.stats.binom

scipy.stats.binom(*args, **kwds) = <scipy.stats._discrete_distns.binom_gen object> [source]

A binomial discrete random variable.

As an instance of the rv_discrete class, binom object inherits from it a collection of generic methods (see below for the full list), and completes them with details specific for this particular distribution.

Notes

The probability mass function for binom is:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for `k` in `{0, 1,..., n}`.

Methods available in binom module

Methods

rvs(n, p, loc=0, size=1, random_state=None)	Random variates.
pmf(k, n, p, loc=0)	Probability mass function.
logpmf(k, n, p, loc=0)	Log of the probability mass function.
cdf(k, n, p, loc=0)	Cumulative distribution function.
logcdf(k, n, p, loc=0)	Log of the cumulative distribution function.
sf(k, n, p, loc=0)	Survival function (also defined as `1 - cdf`, but *sf* is sometimes more accurate).
logsf(k, n, p, loc=0)	Log of the survival function.
ppf(q, n, p, loc=0)	Percent point function (inverse of `cdf` — percentiles).
isf(q, n, p, loc=0)	Inverse survival function (inverse of `sf`).
stats(n, p, loc=0, moments='mv')	Mean('m'), variance('v'), skew('s'), and/or kurtosis('k').
entropy(n, p, loc=0)	(Differential) entropy of the RV.
expect(func, args=(n, p), loc=0, lb=None, ub=None, conditional=False)	Expected value of a function (of one argument) with respect to the distribution.
median(n, p, loc=0)	Median of the distribution.
mean(n, p, loc=0)	Mean of the distribution.
var(n, p, loc=0)	Variance of the distribution.
std(n, p, loc=0)	Standard deviation of the distribution.
interval(alpha, n, p, loc=0)	Endpoints of the range that contains alpha percent of the distribution

Binomial Distribution in Python

- Generating Number from Binomial Distribution

```
from scipy.stats import binom
```

```
data=binom.rvs(n=6,p=0.5,size=100)
data
```

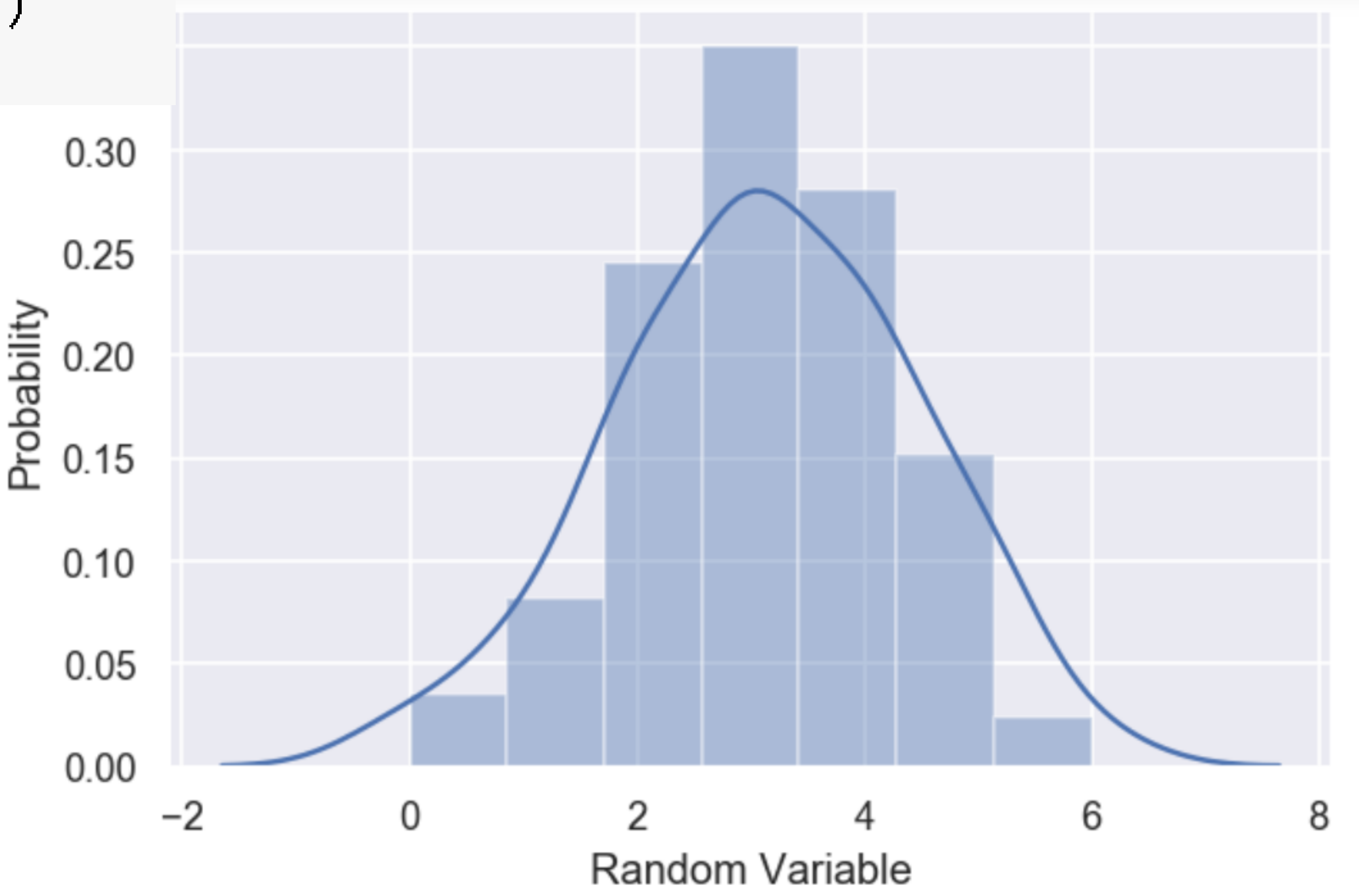
```
array([3, 0, 3, 4, 4, 3, 2, 2, 1, 2, 3, 3, 4, 3, 5, 5, 3, 4, 2, 4, 3, 4,
       5, 2, 5, 3, 0, 1, 4, 5, 3, 3, 2, 4, 1, 4, 5, 4, 2, 1, 4, 4, 2, 2,
       3, 5, 2, 3, 4, 3, 6, 2, 3, 3, 4, 1, 4, 4, 3, 4, 4, 5, 4, 2, 3, 3,
       2, 3, 3, 3, 3, 5, 3, 3, 2, 2, 4, 3, 2, 4, 5, 2, 4, 3, 4, 2, 1, 5,
       2, 2, 5, 5, 0, 3, 4, 3, 1, 3, 6, 2])
```

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

```
(array([0, 1, 2, 3, 4, 5, 6]),
 array([ 3,  7, 21, 30, 24, 13,  2], dtype=int64))
```

Plotting the Binomial Distribution

```
plt.figure(dpi=120)
sns.distplot(data)
plt.xlabel("Random Variable")
plt.ylabel("Probability")
plt.show()
```



Estimation of CDF and its inverse

In [53]:
```
binom.cdf(k=3,n=6,p=0.7)
```
Out[53]: 0.25569000000000003

In [56]:
```
#Percent point function (inverse of cdf – percentiles).
binom.ppf(q=0.22569, n=6, p=0.7)
```
Out[56]: 3.0

In [60]:
```
# WHat should be the k value if I want probability to be .80
binom.ppf(q=0.80,n=6,p=0.7)
```
Out[60]: 5.0