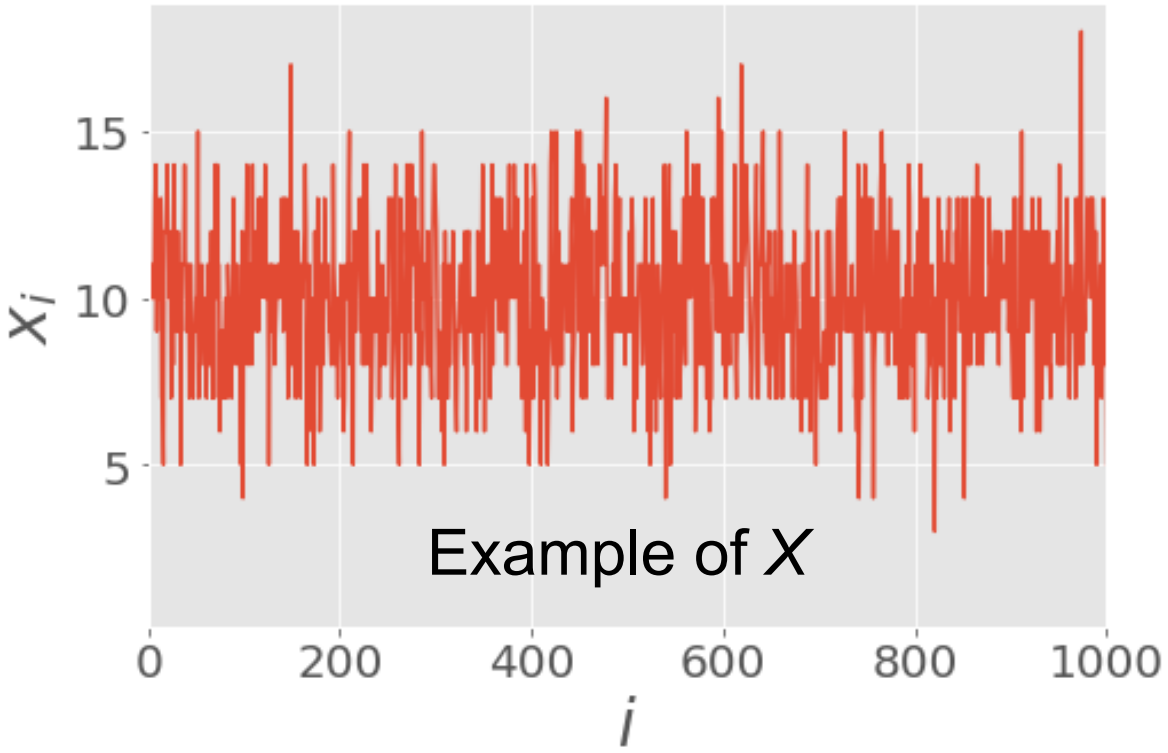
Distribution functions & random numbers

Stochastic variables and distribution functions

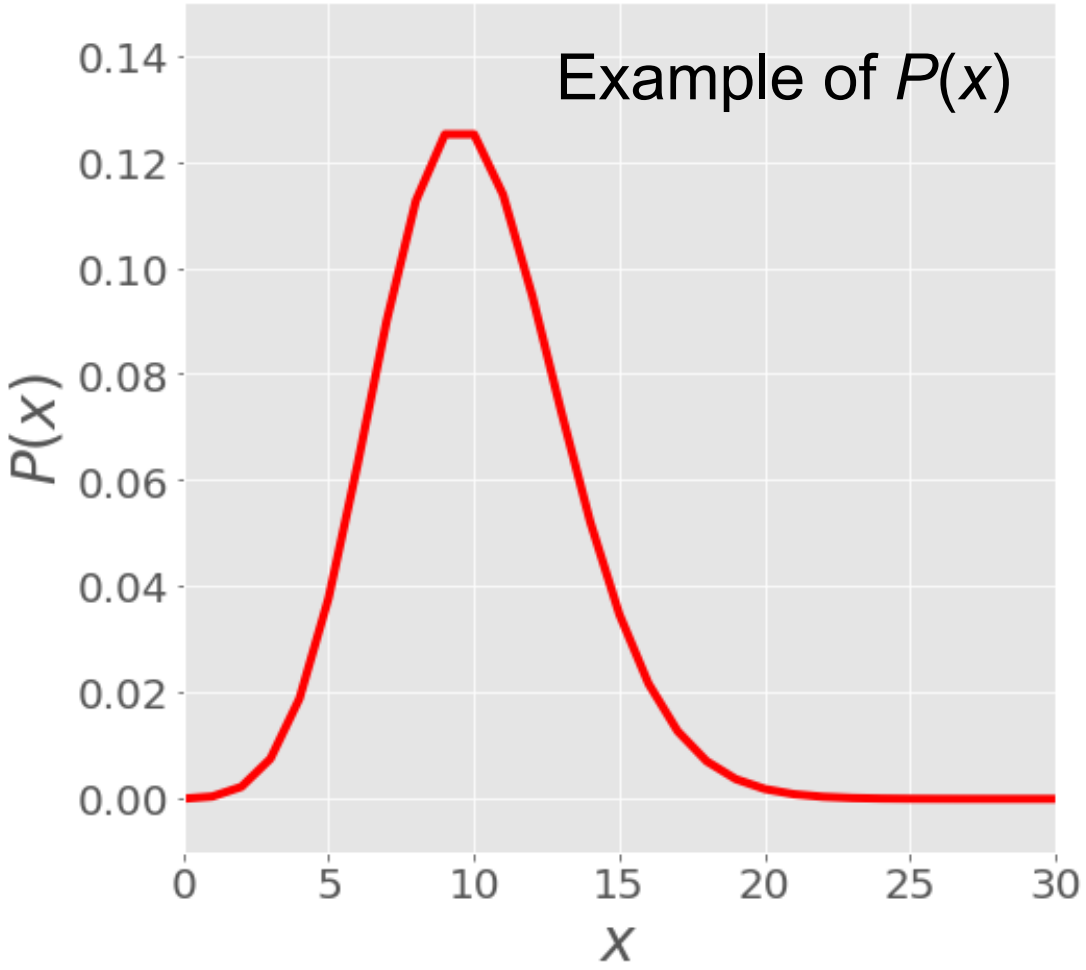
Stochastic variable and distribution functions

Stochastic variable:

$$X(=x_1, x_2, \cdots)$$



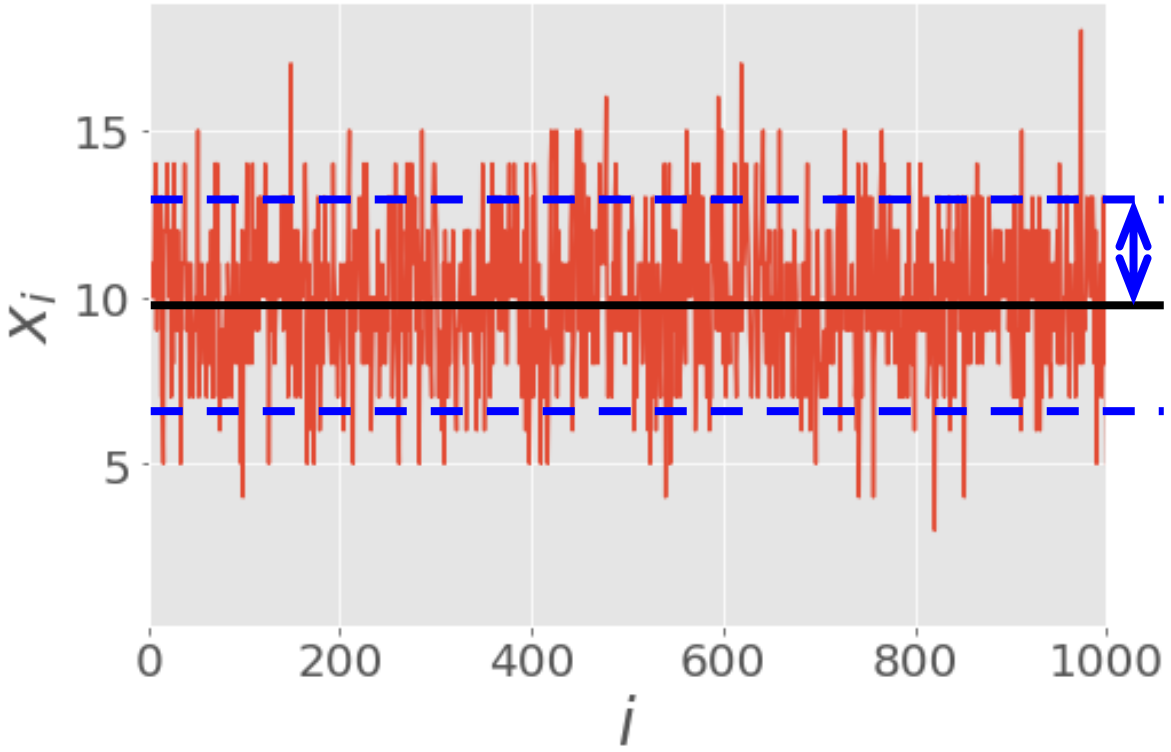
Distribution function: $P(x)$



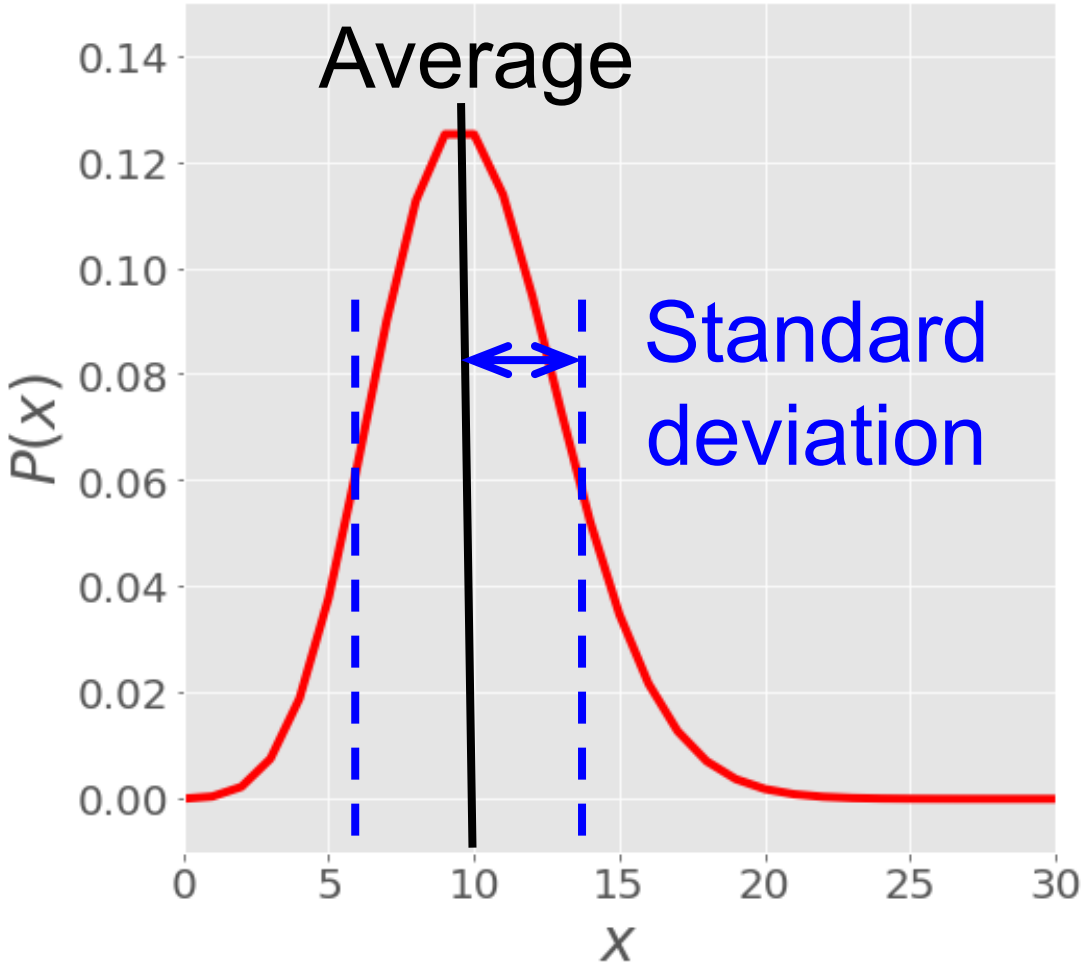
Stochastic variable and distribution functions

Stochastic variable:

$$X(=x_1, x_2, \cdots)$$



Distribution function: $P(x)$



Stochastic variable and distribution functions

Basic things to know about $P(x)$ for x = real & continuum

1. Positive : $P(x) \geq 0$

2. Normalization : $\int_{-\infty}^{\infty} P(x)\,dx = 1$

3. Moment (m-th) : $\mu_m \equiv \langle X^m \rangle = \int_{-\infty}^{\infty} x^m P(x)\,dx$

4. Average : $\langle f(X) \rangle = \int_{-\infty}^{\infty} f(x) P(x)\,dx$

$$\langle X \rangle = \int_{-\infty}^{\infty} x P(x)\,dx = \mu_1$$

Stochastic variable and distribution functions

Basic things to know about $P(x)$ for x = real & continuum

5. Variance : $\sigma^2 \equiv \left\langle \left(X - \langle X \rangle \right)^2 \right\rangle = \langle X^2 \rangle - \langle X \rangle^2 = \mu_2 - \mu_1^2$

(σ is the Standard deviation)

6. Generating function : $G(k) \equiv \left\langle e^{-ikx} \right\rangle = \sum_{n=0}^{\infty} (ik)^n \frac{\mu_n}{n!}$

$$G'(0) = \mu_1, \quad G''(0) = \mu_2/2!, \quad G'''(0) = \mu_3/3!, \quad \cdots$$

$$\left(\because e^{-ikx} = 1 - ikx + \frac{1}{2!}(-ik)^2 x^2 + \frac{1}{3!}(-ik)^3 x^3 + \cdots \right)$$

Stochastic variable and distribution functions

Basic things to know about $P(n)$ for n = non-negative integer

1. Positive : $0 \le P(n) \le 1$
2. Normalization : $\sum_{n=0}^{\infty} P(n) = 1$
3. Moment (m-th) : $\mu_m \equiv \langle N^m \rangle = \sum_{n=0}^{\infty} n^m P(n)$
4. Average : $\langle f(N) \rangle = \sum_{n=0}^{\infty} f(n) P(n)$

$$\langle N \rangle = \sum_{n=0}^{\infty} n P(n) = \mu_1$$

Stochastic variable and distribution functions

Basic things to know about $P(n)$ for n = non-negative integer

5. Variance : $\sigma^2 \equiv \left\langle \left(N - \langle N \rangle \right)^2 \right\rangle = \langle N^2 \rangle - \langle N \rangle^2 = \mu_2 - \mu_1^2$

6. Generating function :

$$G(k) \equiv \left\langle e^{-ikn} \right\rangle = \sum_{n=0}^{\infty} e^{-ikn} P(n)$$

$$G(z \equiv e^{-ik}) = \sum_{n=0}^{\infty} z^n P(n)$$

$$G(z)\Big|_{z=1} = 1, \quad G'(z)\Big|_{z=1} = \mu_1$$

$$G''(z)\Big|_{z=1} = \left\langle N(N-1) \right\rangle, \quad \cdots$$

Stochastic variable and distribution functions

Normal distribution = Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi s^2}} \exp\left[-\frac{(x - x_0)^2}{2s^2} \right] \qquad \text{(C1)}$$

$$\mu_1 \equiv \langle X \rangle = x_0 \qquad \text{(C2)}$$

$$\sigma^2 \equiv \langle X^2 \rangle - \langle X \rangle^2 = s^2 \qquad \text{(C3)}$$

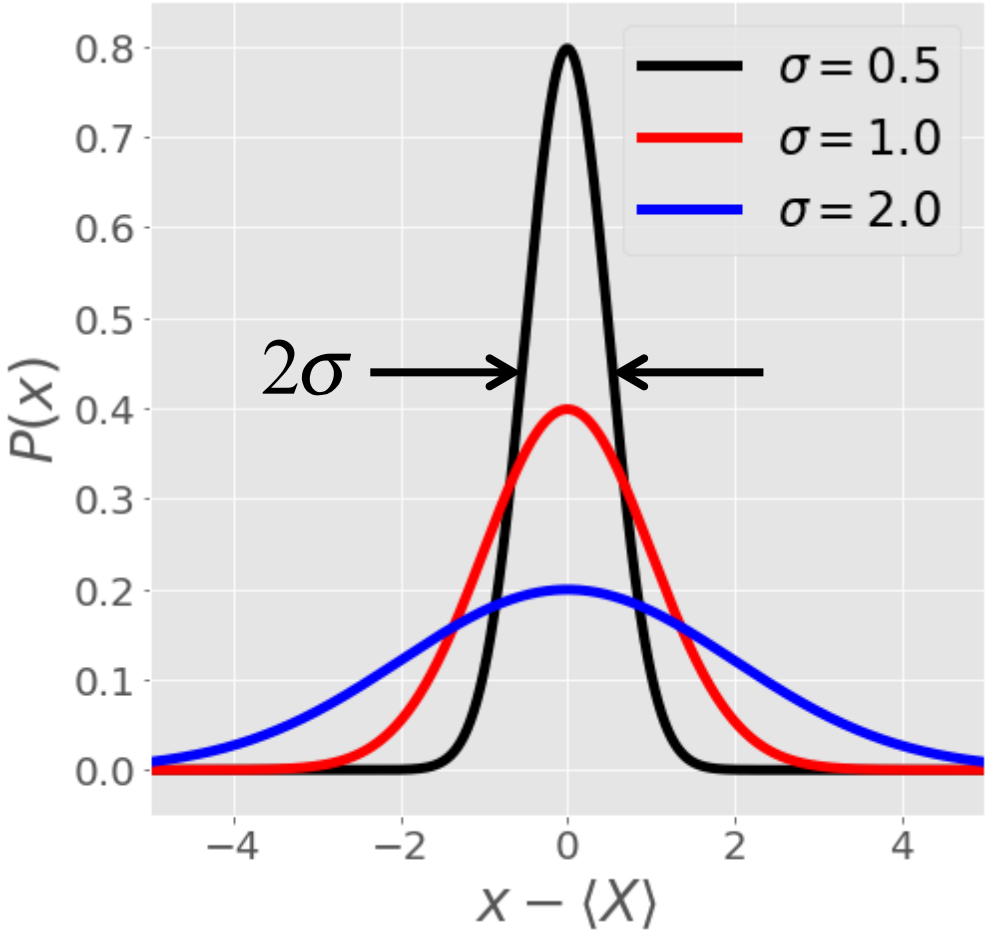
Stochastic variable and distribution functions

Normal distribution = Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi s^2}} \exp\left[-\frac{(x - x_0)^2}{2s^2} \right] \qquad \text{(C1)}$$

$$\mu_1 \equiv \langle X \rangle = x_0 \qquad \text{(C2)}$$

$$\sigma^2 \equiv \langle X^2 \rangle - \langle X \rangle^2 = s^2 \qquad \text{(C3)}$$



Stochastic variable and distribution functions

Ex. Maxwell-Boltzmann distribution

- Velocity of molecules : $V_\alpha \quad (\alpha = x, y, z)$
- Mass of molecule : m
- Temperature : T
- Boltzmann constant : k_B

$$P(v_\alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{v_\alpha{}^2}{2\sigma^2} \right] \qquad \text{(C4)}$$

$$\langle V_\alpha \rangle = 0, \quad \langle V_\alpha^2 \rangle = \sigma^2 = \frac{k_B T}{2m} \qquad \text{(C5)}$$

Stochastic variable and distribution functions

Binomial distribution

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \qquad \text{(C6)}$$

$$\mu_1 = \sum_{n=0}^{M} nP(n) = Mp \qquad \text{(C7)}$$

$$\sigma^2 = Mp(1-p) \qquad \text{(C8)}$$

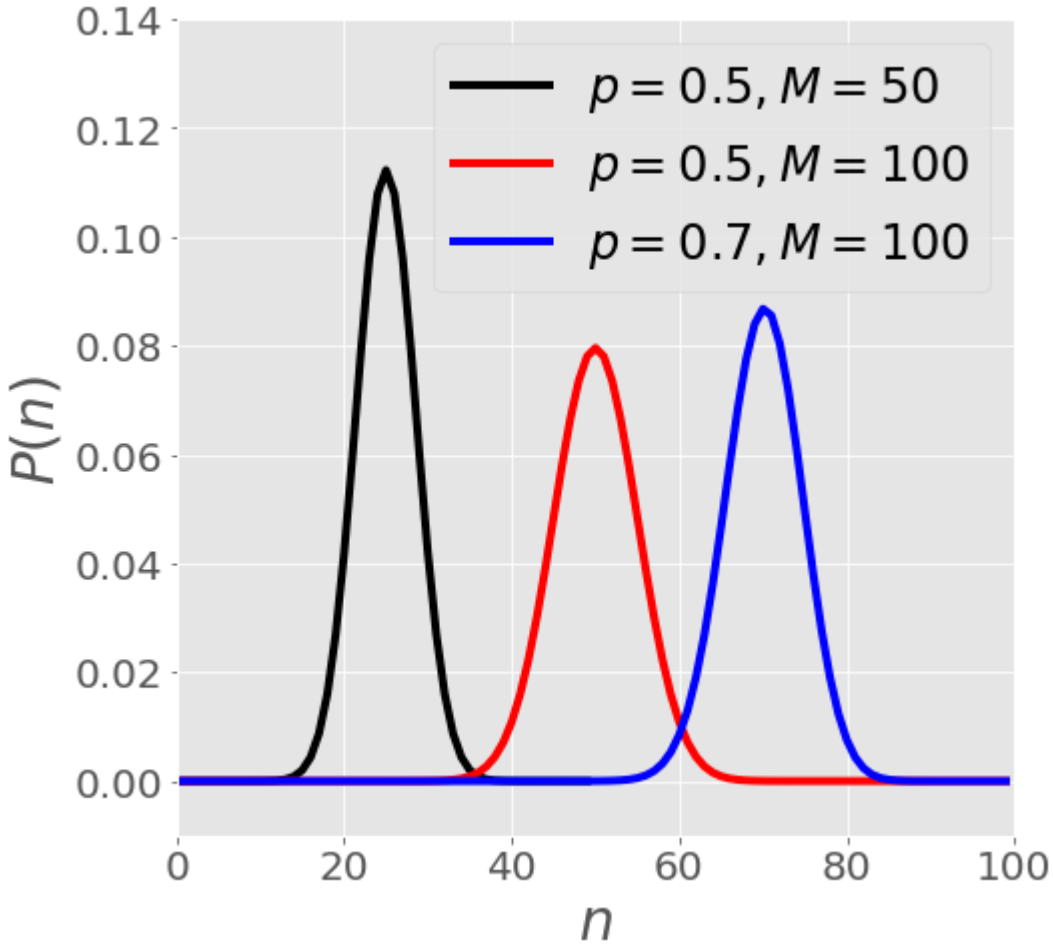
Stochastic variable and distribution functions

Binomial distribution

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \qquad \text{(C6)}$$

$$\mu_1 = \sum_{n=0}^{M} nP(n) = Mp \qquad \text{(C7)}$$

$$\sigma^2 = Mp(1-p) \qquad \text{(C8)}$$



Stochastic variable and distribution functions

Poisson distribution

$$P(n) = \frac{a^n e^{-a}}{n!} \qquad \text{(C9)}$$

$$G(z) = e^{a(z-1)} \qquad \text{(C10)}$$

$$\mu_1 = G'(z)\Big|_{z=0} = a \qquad \text{(C11)}$$

$$\sigma^2 = \left\langle N^2 \right\rangle - \left\langle N \right\rangle^2 = a \qquad \text{(C12)}$$

$$\left(\because \left\langle N(N-1) \right\rangle = G''(z)\Big|_{z=0} = a^2 \right) \qquad \text{(C13)}$$

Stochastic variable and distribution functions

Poisson distribution

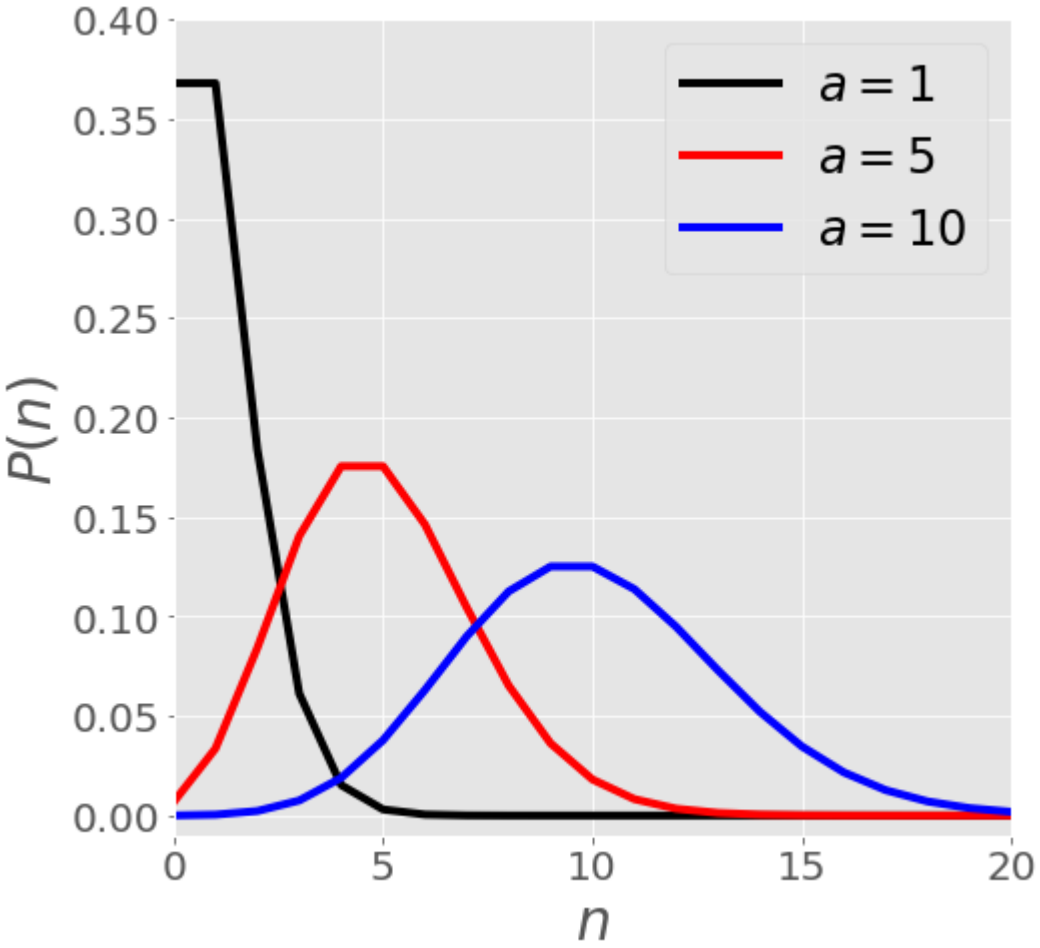
$$P(n) = \frac{a^n e^{-a}}{n!} \tag{C9}$$

$$G(z) = e^{a(z-1)} \tag{C10}$$

$$\mu_1 = G'(z)\Big|_{z=0} = a \tag{C11}$$

$$\sigma^2 = \left\langle N^2 \right\rangle - \left\langle N \right\rangle^2 = a \tag{C12}$$

$$\left(\because \left\langle N(N-1) \right\rangle = G''(z)\Big|_{z=0} = a^2 \right) \tag{C13}$$



Stochastic variable and distribution functions

Binomial distribution $\rightarrow$ Normal distribution

See supplemental note for derivation.

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \qquad \text{(C6) Binomial}$$

$$\xrightarrow[n \rightarrow \textbf{cont.}]{n, M \gg 1} P(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(n-\mu_1)^2}{2\sigma^2} \right] \qquad \text{(C1) Normal}$$

$$\mu_1 \equiv \langle N \rangle = Mp \qquad \text{(C7)}$$

$$\sigma^2 \equiv \langle N^2 \rangle - \langle N \rangle^2 = Mp(1-p) \qquad \text{(C8)}$$

Stochastic variable and distribution functions

Binomial distribution $\rightarrow$ Poisson distribution

See supplemental note for derivation.

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \qquad \text{(C6) Binomial}$$

$$\xrightarrow[Mp = a = \textbf{const.}]{M \gg 1} \quad P(n) = \frac{a^n e^{-a}}{n!} \qquad \text{(C9) Poisson}$$

$$\mu_1 \equiv a = Mp \qquad \text{(C11)}$$

$$\sigma^2 \equiv a = Mp(1-p) \qquad \text{(C12)}$$

$$\simeq Mp \qquad \text{(C14)}$$

Stochastic Processes:
Data Analysis and Computer Simulation

Distribution function and random number

2. Generating random numbers with Gaussian/binomial/Poisson distributions

2.1. Preparations

Python built-in functions for random numbers (numpy.random)

1. `seed(seed)`: Initialize the generator with an integer "seed".
2. `rand(d0,d1,...,dn)`: Return a multi-dimensional array of uniform random numbers of shape (d0, d1, ... , dn).
3. `randn(d0,d1,...,dn)`: The same as above but from the standard normal distribution.
4. `binomial(M,p,size)`: Draw samples from a binomial distribution with "M" and "p".
5. `poisson(a,size)`: Draw samples from a Poisson distribution with "a".
6. `choice([-1,1],size)`: Generates random samples from the two choices, -1 or 1 in this case.
7. `normal(ave,std,size)`: Draw random samples from a normal distribution.
8. `uniform([low,high,size])`: Draw samples from a uniform distribution.

- See the Scipy website for details
  https://docs.scipy.org/doc/numpy-dev/reference/routines.random.html
  (https://docs.scipy.org/doc/numpy-dev/reference/routines.random.html)

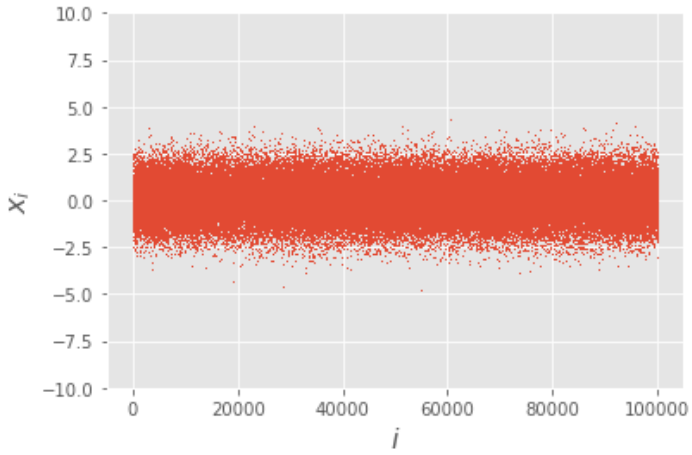
Import common libraries

In [1]:
```
% matplotlib inline
import numpy as np # import numpy library as np
import math # use mathematical functions defined by the C standard
import matplotlib.pyplot as plt # import pyplot library as plt
plt.style.use('ggplot') # use "ggplot" style for graphs
```

2.2. Normal / Gaussian distribution

Generate random numbers, $x_0, x_1, \cdots, x_N$

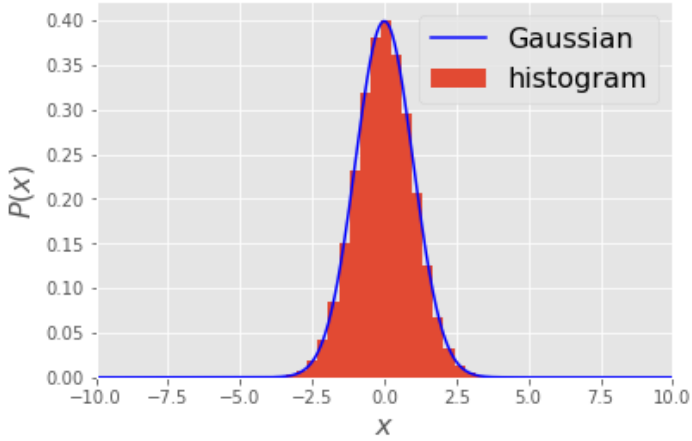
In [2]:
```
ave = 0.0 # set average
std = 1.0 # set standard deviation
N = 100000 # number of generated random numbers
np.random.seed(0) # initialize the random number generator with seed=0
X = ave+std*np.random.randn(N) # generate random sequence and store it a
plt.ylim(-10,10) # set y-range
plt.xlabel(r'$i$',fontsize=16) # set x-label
plt.ylabel(r'$x_i$',fontsize=16) # set y-label
plt.plot(X,',') # plot x_i vs. i (i=1,2,...,N) with dots
plt.show() # draw plots
```



Compare the distribution with the normal distribution function

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x - \langle X \rangle)^2}{2\sigma^2}\right] \tag{D1}$$

In [3]:
```
plt.hist(X,bins=25,normed=True) # plot normalized histogram of R using 2
x = np.arange(-10,10,0.01) # create array of x from 0 to 1 with incremen
y = np.exp(-(x-ave)**2/(2*std**2))/np.sqrt(2*np.pi*std**2) # create arra
plt.xlim(-10,10) # set x-range
plt.plot(x,y,color='b') # plot y vs. x with blue line
plt.xlabel(r'$x$',fontsize=16) # set x-label
plt.ylabel(r'$P(x)$',fontsize=16) # set y-label
plt.legend([r'Gaussian',r'histogram'], fontsize=16) # set legends
plt.show() # display plots
```



Calculate the auto-correlation function $\varphi(i)$

The definition

$$\varphi(i) = \frac{1}{N}\sum_{j=1}^{N}\left(x_j - \langle X\rangle\right)\left(x_{i+j} - \langle X\rangle\right) \tag{D2}$$

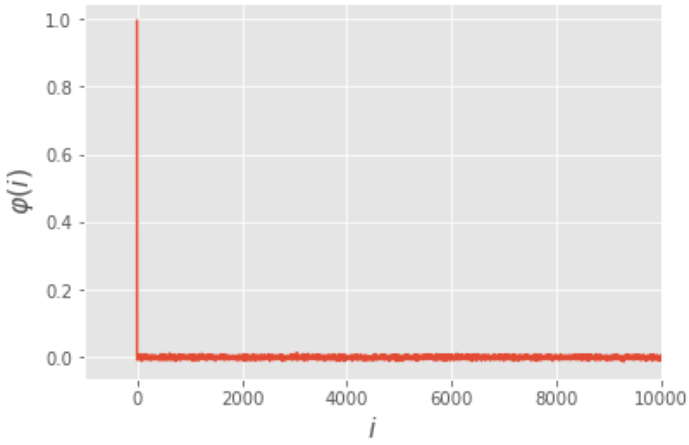
$$\varphi(i=0) = \frac{1}{N}\sum_{j=1}^{N}\left(x_j - \langle X\rangle\right)^2 = \langle x_j - \langle X\rangle\rangle^2 = \sigma^2 \tag{D3}$$

$$\varphi(i\neq 0) = \langle x_j - \langle X\rangle\rangle\langle x_{i\neq j} - \langle X\rangle\rangle = 0 \quad (\rightarrow \text{White noise}) \tag{D4}$$

A code example to calculate auto-correlation

In [4]:
```
def auto_correlate(x):
    cor = np.correlate(x,x,mode="full")
    return cor[N-1:]
c = np.zeros(N)
c = auto_correlate(X-ave)/N
plt.plot(c)
plt.xlim(-1000,10000)
plt.xlabel(r'$i$',fontsize=16)
plt.ylabel(r'$\varphi(i)$',fontsize=16)
print('\sigma^2  =',std**2)
plt.show()
```

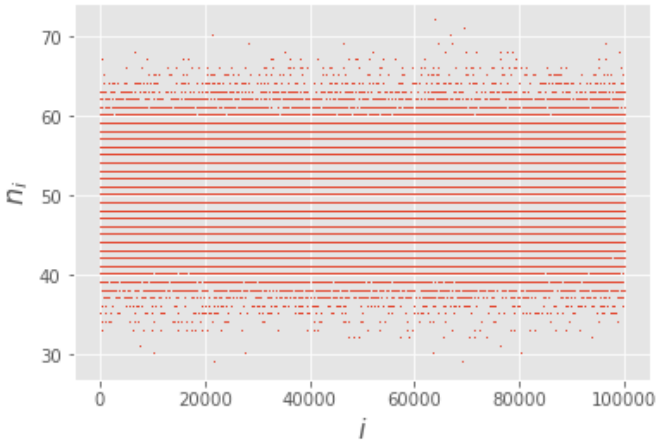
```
\sigma^2  = 1.0
```



2.3. Binomial distribution

Generate random numbers, $n_0, n_1, \cdots, n_N$

In [5]:
```
p = 0.5 # set p, propability to obtain "head" from a coin toss
M = 100 # set M, number of tosses in one experiment
N = 100000 # number of experiments
np.random.seed(0) # initialize the random number generator with seed=0
X = np.random.binomial(M,p,N) # generate the number of heads after M tos
plt.xlabel(r'$i$',fontsize=16) # set x-label
plt.ylabel(r'$n_i$',fontsize=16) # set y-label
plt.plot(X,',') # plot n_i vs. i (i=1,2,...,N) with dots
plt.show() # draw plots
```



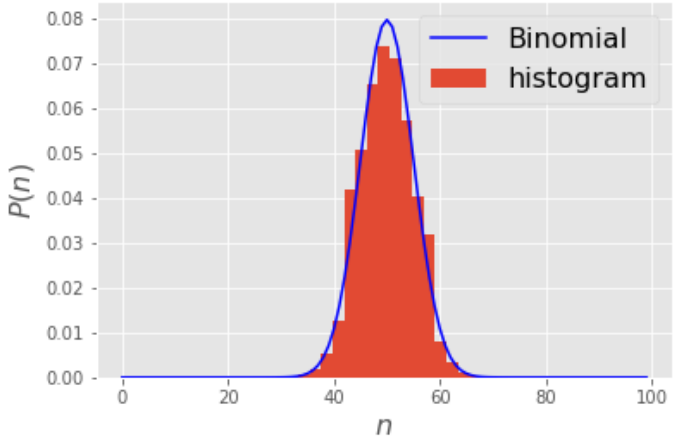
Compare the distribution with the Binomial distribution function

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \tag{D5}$$

$$\langle n \rangle = Mp \tag{D6}$$

$$\sigma^2 = Mp(1-p) \tag{D7}$$

In [6]:
```
def binomial(n,m,p):
    comb=math.factorial(m)/(math.factorial(n)*math.factorial(m-n))
    prob=comb*p**n*(1-p)**(m-n)
    return prob
plt.hist(X,bins=20,normed=True) # plot normalized histogram of R using 2
x = np.arange(M) # generate array of x values from 0 to 100, in interval
y = np.zeros(M) # generate array of y values, initialized to 0
for i in range(M):
    y[i]=binomial(i,M,p) # compute binomial distribution P(n), Eq. (D5)
plt.plot(x,y,color='b') # plot y vs. x with blue line
plt.xlabel(r'$n$',fontsize=16) # set x-label
plt.ylabel(r'$P(n)$',fontsize=16) # set y-label
plt.legend([r'Binomial',r'histogram'], fontsize=16) # set legends
plt.show() # display plots
```



Calculate the auto-correlation function $\varphi(i)$

The definition

$$\varphi(i) = \frac{1}{N} \sum_{j=1}^{N} \left(n_j - \langle n \rangle\right) \left(n_{i+j} - \langle n \rangle\right) \tag{D8}$$

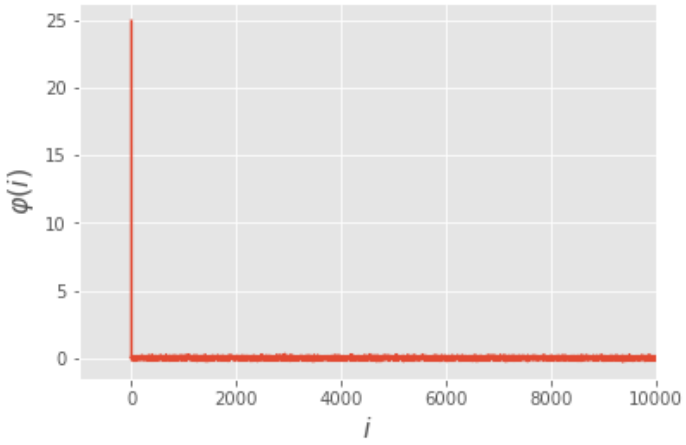
$$\varphi(i=0) = \frac{1}{N} \sum_{j=1}^{N} \left(n_j - \langle n \rangle\right)^2 = \langle n_j - \langle n \rangle \rangle^2 = \sigma^2 = Mp(1-p) \tag{D9}$$

$$\varphi(i \neq 0) = \langle n_j - \langle n \rangle \rangle \langle n_{i \neq j} - \langle n \rangle \rangle = 0 \qquad (\rightarrow \text{White noise}) \tag{D10}$$

A code example to calculate auto-correlation

In [7]:
```
def auto_correlate(x):
    cor = np.correlate(x,x,mode="full")
    return cor[N-1:]
c = np.zeros(N)
c = auto_correlate(X-M*p)/N
plt.plot(c)
plt.xlim(-1000,10000)
plt.xlabel(r'$i$',fontsize=16)
plt.ylabel(r'$\varphi(i)$',fontsize=16)
print('\sigma^2  =',M*p*(1-p))
plt.show()
```

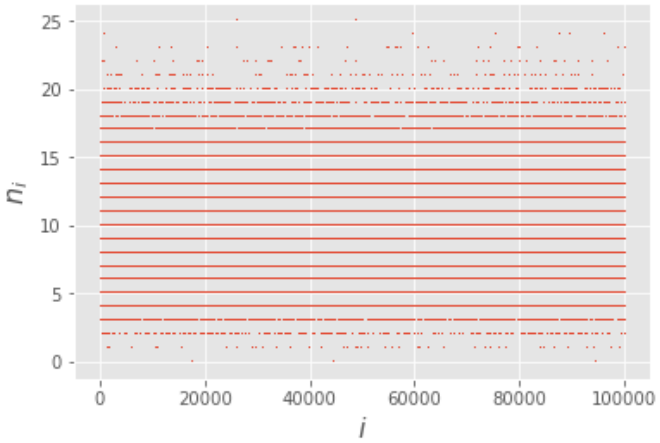
```
\sigma^2  = 25.0
```



2.4. Poisson distribution

Generate random numbers, $n_0, n_1, \cdots, n_N$

In [8]:
```
a = 10.0 # set a, the expected value
N = 100000 #  number of generated random numbers
np.random.seed(0) # initialize the random number generator with seed=0
X = np.random.poisson(a,N) # generate randon numbers from poisson distri
plt.xlabel(r'$i$',fontsize=16) # set x-label
plt.ylabel(r'$n_i$',fontsize=16) # set y-label
plt.plot(X,',') # plot n_i vs. i (i=1,2,...,N) with dots
plt.show() # draw plots
```



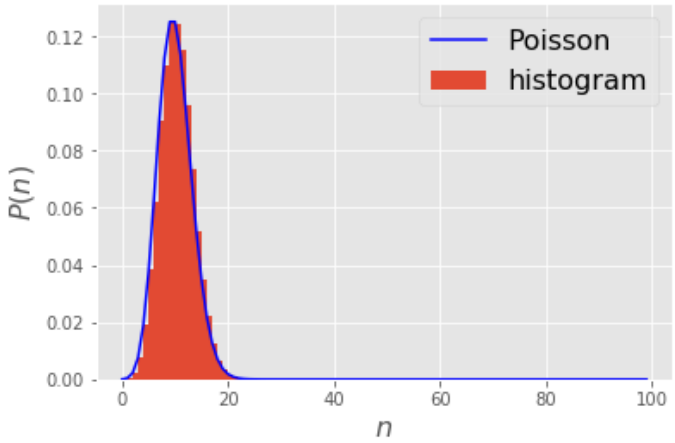
Compare the distribution with the binomial distribution function

$$P(n) = \frac{a^n e^{-a}}{n!} \tag{D11}$$

$$\langle n \rangle = a \tag{D12}$$

$$\sigma^2 = a \tag{D13}$$

In [9]:
```
def poisson(n,a):
    prob=a**n*np.exp(-a)/math.factorial(n)
    return prob
plt.hist(X,bins=25,normed=True) # plot normalized histogram of X using 2
x = np.arange(M) # generate array of x values from 0 to 100, in interval
y = np.zeros(M) # generate array of y values, initialized to zero
for i in range(M):
    y[i]=poisson(i,a) # Compute Poisson distribution for n, Eq. (D11)
plt.plot(x,y,color='b') # plot y vs. x with blue line
plt.xlabel(r'$n$',fontsize=16) # set x-label
plt.ylabel(r'$P(n)$',fontsize=16) # set y-label
plt.legend([r'Poisson',r'histogram'], fontsize=16) # set legends
plt.show() # display plots
```



Stochastic Processes:
Data Analysis and Computer Simulation

Distribution function and random number

3. The central limit theorem

3.1. Binomial distribution → Gauss distribution

From the previous lesson

- The binomial distribution becomes equivalent to the Gaussian distribution in the limit $n, M \gg 1$, as shown in the 1st plot of this week.

$$P(n) = \frac{M!}{n!(M-n)!} p^n (1-p)^{M-n} \tag{C6}$$

$$\xrightarrow[n \to cont.]{n, M \gg 1} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(n-\mu_1)^2}{2\sigma^2} \right] \tag{C1}$$

$$\mu_1 = Mp, \quad \sigma^2 = Mp(1-p) \tag{C7, C8}$$

Numerical experiment 1

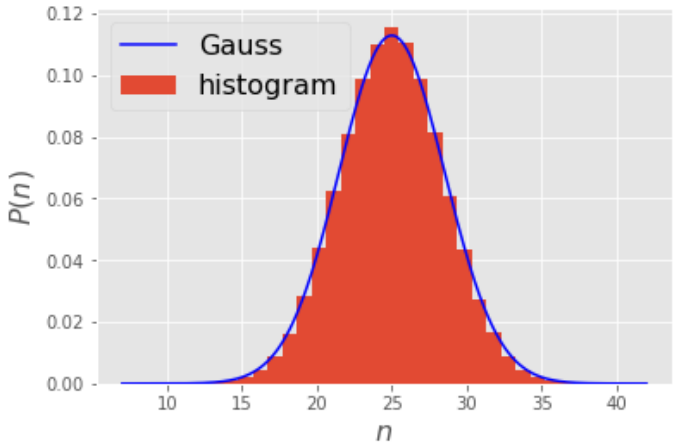
- While the proof for the equivalence has been given in the supplemental note, let us examine this by performing numerical experiments for various values of $M = 1, 2, 4, 10, 100$ and 1000.

Include libraries

In [1]:
```
% matplotlib inline
import numpy as np # import numpy library as np
import math # use mathematical functions defined by the C standard
import matplotlib.pyplot as plt # import pyplot library as plt
plt.style.use('ggplot') # use "ggplot" style for graphs
```

In [2]:
```
p = 0.5    # set p, propability to obtain "head" from a coin toss
M = 50      # set M, number of tosses in one experiment
N = 100000 # number of experiments
ave = M*p
std = np.sqrt(M*p*(1-p))
print('p =',p,'M =',M)
np.random.seed(0) # initialize the random number generator with seed=0
X = np.random.binomial(M,p,N) # generate the number of head come up N ti
nmin=np.int(ave-std*5)
nmax=np.int(ave+std*5)
nbin=nmax-nmin+1
plt.hist(X,range=[nmin,nmax],bins=nbin,normed=True) # plot normalized hi
x = np.arange(nmin,nmax,0.01/std) # create array of x from nmin to nmax
y = np.exp(-(x-ave)**2/(2*std**2))/np.sqrt(2*np.pi*std**2) # calculate t
plt.plot(x,y,color='b') # plot y vs. x with blue line
plt.xlabel(r'$n$',fontsize=16) # set x-label
plt.ylabel(r'$P(n)$',fontsize=16) # set y-label
plt.legend([r'Gauss',r'histogram'], fontsize=16) # set legends
plt.show() # display plots
```

```
p = 0.5 M = 50
```



What we can learn from the experiment

- Stochastic variable "s" is a result of single binary choice,

$$s = 0 \text{ or } 1 \tag{1}$$

and Stochastic variable "n^M" is a sum of M independent binary choices s, with the index j representing the j-th choice.

$$n^M = \sum_{j=1}^{M} s_j \tag{2}$$

For $M = 1$

$$n^{M=1} = s_1 = s = 0 \text{ or } 1 \tag{D1}$$

- Distribution function → Binary choice, $P^{M=1}(0) = 1 - p,\ P^{M=1}(1) = p$, with

$$\mu_1^{M=1} = p, \qquad \sigma_{M=1}^2 = p(1-p) \tag{D2, D3}$$

For $M \gg 1$

$$n^M = \sum_{j=1}^{M} s_j = \sum_{j=1}^{M} n_j^{M=1} \tag{D4}$$

- Distribution function → Gaussian with

$$\mu_1^{M\gg1} = M\mu_1^{M=1}, \qquad \sigma_{M\gg1}^2 = M\sigma_{M=1}^2 \tag{D5, D6}$$

3.2. The central limiting theorem (CLT)

Generalization of Eqs. (D4-D6) for $M \gg 1$

CLT for sum of stochastic variables

- Stochastic variable "n^M" as a SUM of any M independent stochastic variables $n^{M=1}$ with $\mu_1^{M=1}$ and $\sigma_{M=1}^2$,

$$n^M = \sum_{j=1}^{M} n_j^{M=1} \tag{D7}$$

- Distribution function → Gauss with

$$\mu_1^{M\gg1} = M\mu_1^{M=1}, \qquad \sigma_{M\gg1}^2 = M\sigma_{M=1}^2 \tag{D8, D9}$$

CLT for average of stochastic variables

- Stochastic variable "n^M" as an AVERAGE of any M independent stochastic variables with $\mu^{M=1}$ and $\sigma_{M=1}^2$,

$$n^M = \frac{1}{M}\sum_{j=1}^{M} n_j^{M=1} \tag{D10}$$

- Distribution function → Gauss with

$$\mu_1^{M\gg1} = \mu_1^{M=1}, \qquad \sigma_{M\gg1}^2 = \frac{\sigma_{M=1}^2}{M} \tag{D11, D12}$$

- Eqs. (D7-D12) is called "the central limiting theorem".

3.3. Uniform distribution → Gauss distribution

From CLT

For $M = 1$

- Stochastic variable "x" is uniformly distributed between 0 and 1,

$$x^{M=1} \in [0 : 1] \tag{D13}$$

- Distribution function: $P^{M=1}(x) = 1$ (for $0 \leq x < 1$), $P^{M=1}(x) = 0$ (otherwise)

$$\mu_1^{M=1} = \frac{1}{2}, \qquad \sigma^2_{M=1} = \frac{1}{12} \tag{D14, D15}$$

For $M \gg 1$

- Stochastic variable "x" is a sum of M independent uniform random numbers

$$x^M = \sum_{j=1}^{M} x_j^{M=1} \tag{D16}$$

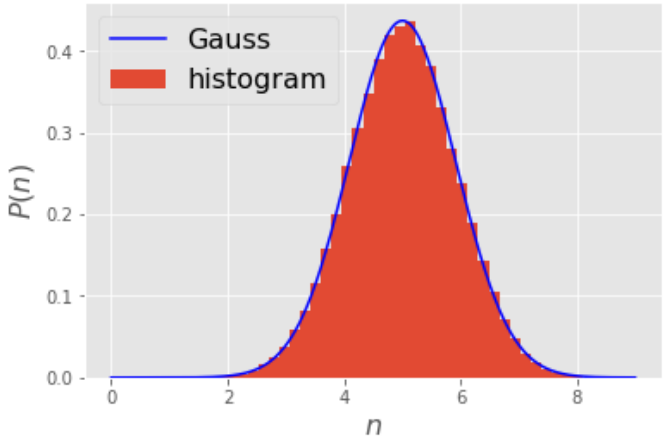
- Distribution function → Gauss with

$$\mu_1^{M\gg1} = M\mu_1^{M=1} = \frac{M}{2}, \qquad \sigma^2_{M\gg1} = M\sigma^2_{M=1} = \frac{M}{12} \tag{D17, D18}$$

Numerical experiment 2

In [3]:
```
M = 10    # set M, the number of random variables to add
N = 100000 # number samples to draw, for each of the random variables
ave = M/2
std = np.sqrt(M/12)
print('M =',M)
np.random.seed(0)        # initialize the random number generator with s
X = np.zeros(N)
for i in range(N):
    X[i] += np.sum(np.random.rand(M)) # draw a random numbers for each c
nmin=np.int(ave-std*5)
nmax=np.int(ave+std*5)
plt.hist(X,range=[nmin,nmax],bins=50,normed=True) # plot normalized hist
x = np.arange(nmin,nmax,0.01/std) # create array of x from nmin to nmax
y = np.exp(-(x-ave)**2/(2*std**2))/np.sqrt(2*np.pi*std**2) # calculate t
plt.plot(x,y,color='b') # plot y vs. x with blue line
plt.xlabel(r'$n$',fontsize=16) # set x-label
plt.ylabel(r'$P(n)$',fontsize=16) # set y-label
plt.legend([r'Gauss',r'histogram'], fontsize=16) # set legends
plt.show() # display plots
```

```
M = 10
```

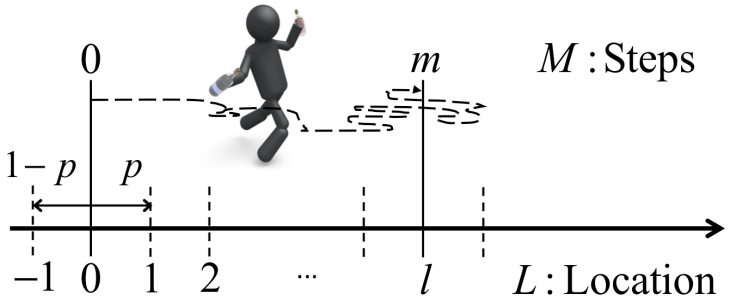


Stochastic Processes: Data Analysis and Computer Simulation

Distribution function and random number

4. Random walk

4.1. The model system (1D random walk)



4.2. As binomial distribution

- The total number of steps to the right: n_+
- The total number of steps to the left: n_-
- The total number of steps: $m = n_+ + n_-$
- The current location: $l = n_+ - n_-$

$$\therefore \quad n_+ = \frac{m+l}{2}, \qquad n_- = \frac{m-l}{2} \tag{E1}$$

$$P(l,m) \to P_{\text{Binomial}}(n_+;m) = P_{\text{Binomial}}(n_-;m) \tag{E2}$$

$$= \frac{m!}{n_+!(m-n_+)!} p^{n_+}(1-p)^{m-n_+} \tag{E3}$$

4.3. As normal distribution (for $n_+, m \gg 1$)

$$P_{\text{Binomial}}(n_+; m) \simeq \frac{1}{\sqrt{2\pi\sigma'^2}} \exp\left[-\frac{(n_+ - \mu'_1)^2}{2\sigma'^2}\right] \tag{E4}$$

$$\text{with} \quad \mu'_1 = \langle n_+ \rangle = mp, \quad \sigma'^2 = \langle n_+^2 \rangle - \langle n_+ \rangle^2 = mp(1-p) \tag{E5, E6}$$

Recall that $n_+ = (m + l) / 2$

$$P_{\text{Binomial}}(n_+; m) \simeq \exp\left[-\frac{(l - m(2p-1))^2}{8mp(1-p)}\right] \tag{E7}$$

$$\therefore \quad P(l, m) = P_{\text{Binomial}}(n_+; m)\frac{dn_+}{dl} = P_{\text{Binomial}}(n_+; m)\frac{1}{2} \tag{E8}$$

$$\simeq \frac{1}{\sqrt{2\pi\sigma''^2}} \exp\left[-\frac{(l - \mu''_1)^2}{2\sigma''^2}\right] \tag{E9}$$

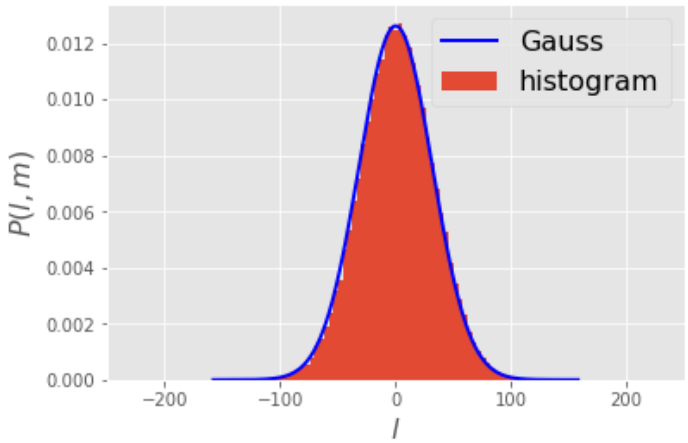
$$\text{with} \quad \mu''_1 = \langle l \rangle = m(2p-1), \quad \sigma''^2 = \langle l^2 \rangle - \langle l \rangle^2 = 4mp(1-p) \tag{E10, E11}$$

4.4. By computer simulation

In [1]:
```
% matplotlib inline
import numpy as np # import numpy library as np
import math # use mathematical functions defined by the C standard
import matplotlib.pyplot as plt # import pyplot library as plt
plt.style.use('ggplot') # use "ggplot" style for graphs
```

In [2]:
```
p = 0.5 # set p, propability to take a step to the right
M = 1000 # M = number of total steps
N = 100000 # N = number of independent random walkers
ave = M*(2*p-1) # average of the location L after M steps Eq.(E10)
std = np.sqrt(4*M*p*(1-p)) # standard deviation of L after M steps Eq.(E
print('p =',p,'M =',M)
L = np.zeros(N)
np.random.seed(0) # initialize the random number generator with seed=0
for i in range(N): # repeat independent random walks N times
    step=np.random.choice([-1,1],M) # generate random sampling from -1 c
    L[i]=np.sum(step) # calculate l after making M random steps and stor
nmin=np.int(ave-std*5)
nmax=np.int(ave+std*5)
nbin=np.int((nmax-nmin)/4)
plt.hist(L,range=[nmin,nmax],bins=nbin,normed=True) # plot normalized hi
x = np.arange(nmin,nmax,0.01/std) # create array of x from nmin to nmax
y = np.exp(-(x-ave)**2/(2*std**2))/np.sqrt(2*np.pi*std**2) # calculate t
plt.plot(x,y,lw=2,color='b') # plot y vs. x with blue line
plt.xlabel(r'$l$',fontsize=16) # set x-label
plt.ylabel(r'$P(l,m)$',fontsize=16) # set y-label
plt.legend([r'Gauss',r'histogram'], fontsize=16) # set legends
plt.xlim(ave-250,ave+250) # set x-range
plt.show() # display plots
```

```
p = 0.5 M = 1000
```



- You may repeat the same simulation by choosing different values of total steps, for example $M = 100, 1,000, 10,000$, and $100,000$ to see how the distribution changes with the total number of steps.

4.5. Connection with the diffusion constant D

$P(x,t)$ from random walk

- Define a as the length of a single step and t_s as the time between subsequent steps.
- Define $x = al$ as the position of the random walker and $t = t_s m$ as the duration of time needed to take m steps.
- Here we consider a drift free case $p = 0.5$, i.e., $\mu_1 = \langle l \rangle = m(2p-1) = 0$.

$$P(x,t) = P(l,m)\frac{dl}{dx} = P(l,m)\frac{1}{a} \tag{E12}$$

$$= \frac{1}{a\sqrt{8\pi mp(1-p)}}\exp\left[-\frac{l^2}{8mp(1-p)}\right] \tag{E13}$$

$$= \frac{1}{\sqrt{8\pi a^2 p(1-p)t/t_s}}\exp\left[-\frac{x^2}{8a^2p(1-p)t/t_s}\right] \tag{E14}$$

$$\text{with} \quad \mu_1 = \langle x \rangle = 0, \;\; \sigma^2 = \langle x^2 \rangle - \langle x \rangle^2 = 4a^2p(1-p)t/t_s \tag{E15, E16}$$

$P(x,t)$ from the diffusion equation

- Consier the 1-D diffusion equation with diffusion constant D

$$\frac{\partial}{\partial t}P(x,t) = D\frac{\partial^2}{\partial x^2}P(x,t) \tag{E17}$$

$$\text{with} \quad P(x, t=0) = \delta(x) \tag{E18}$$

- The solution is given by

$$P(x,t) = \frac{1}{\sqrt{4\pi Dt}}\exp\left[-\frac{x^2}{4Dt}\right] \tag{E19}$$

- By comparing Eqs.(E14) and (E19) we can relate the diffusion constant D to the variance of the position of random walkers

$$D = \frac{2a^2p(1-p)}{t_s} = \frac{\sigma^2}{2t} \tag{E20}$$

- In this case, σ^2 is also referred to as the mean-square displacement