Lab 7: Probability Distributions

Austin Jones and Paige Brady

February 24, 2020

[48]:
```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

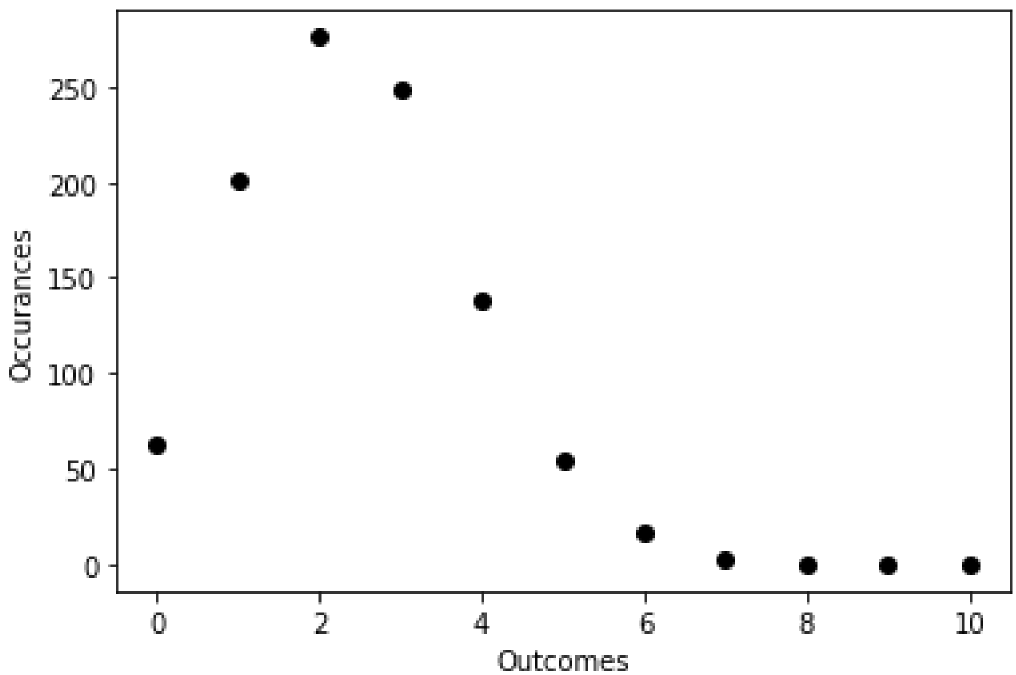
[56]:
```
#7.2 Simulating the Binomial Process

nexp = 1000
ntry = 10
eps = .25

# nexp = int(input("Enter number of experiments: "))
# ntry = int(input("Enter number of trials: "))
# eps = float(input("Enter a value epsilon () < 1: "))

def th_bi(nexp, ntry, eps):
    x = np.random.uniform(size=(nexp, ntry))
    xresult = np.array(x<eps, dtype = int)
    s = np.sum(xresult, axis = 1)
    return s

v = th_bi(nexp, ntry, eps)

print("mean expected:  ", ntry*eps)
print("mean simulated: ", np.mean(v))
print("var expected: ", ntry*eps*(1.0 - eps))
print("var simulated: ", np.var(v))
```

```
mean expected:   2.5
mean simulated:  2.442
var expected:  1.875
var simulated:  1.890636
```

[57]:
```
#7.3 Histogram for the Binomal Process

counts, edges = np.histogram(v, bins = 11, range = (0,11))
print("Counts: ", counts)
print("Total:  ", np.sum(counts))
```

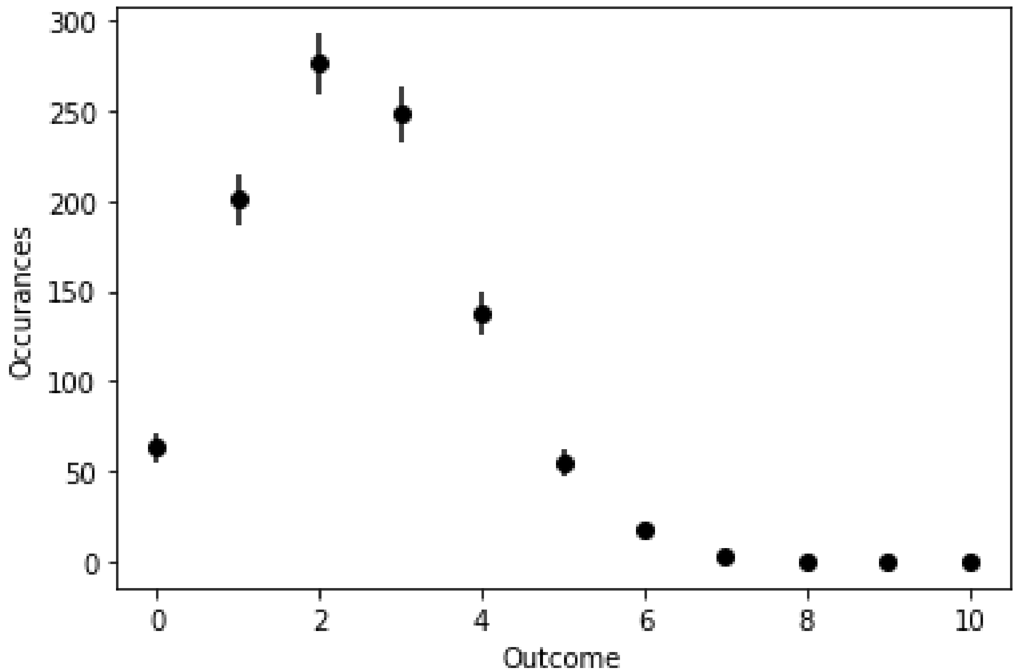
```
print("Bin Edges: ", edges)

plt.plot(edges[:-1], counts, "ko")
plt.xlabel("Outcomes")
plt.ylabel("Occurances")
print("edges[:-1]: ", edges[:-1])
```

```
Counts:  [ 63 201 276 248 138  54  17   3   0   0   0]
Total:   1000
Bin Edges:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11.]
edges[:-1]:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```



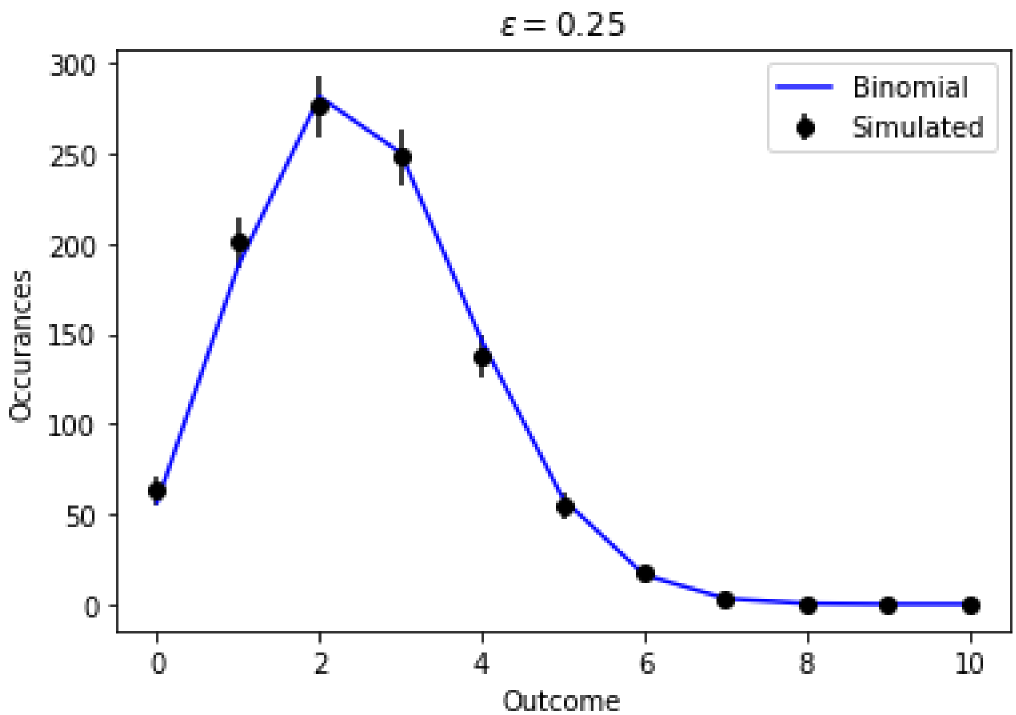
[58]:
```
#7.4 Error Bars

errs = counts**0.5
plt.errorbar(edges[:-1], counts, yerr = errs, fmt = "ko")
plt.xlabel("Outcome")
plt.ylabel("Occurances")
```

[58]:
```
Text(0, 0.5, 'Occurances')
```



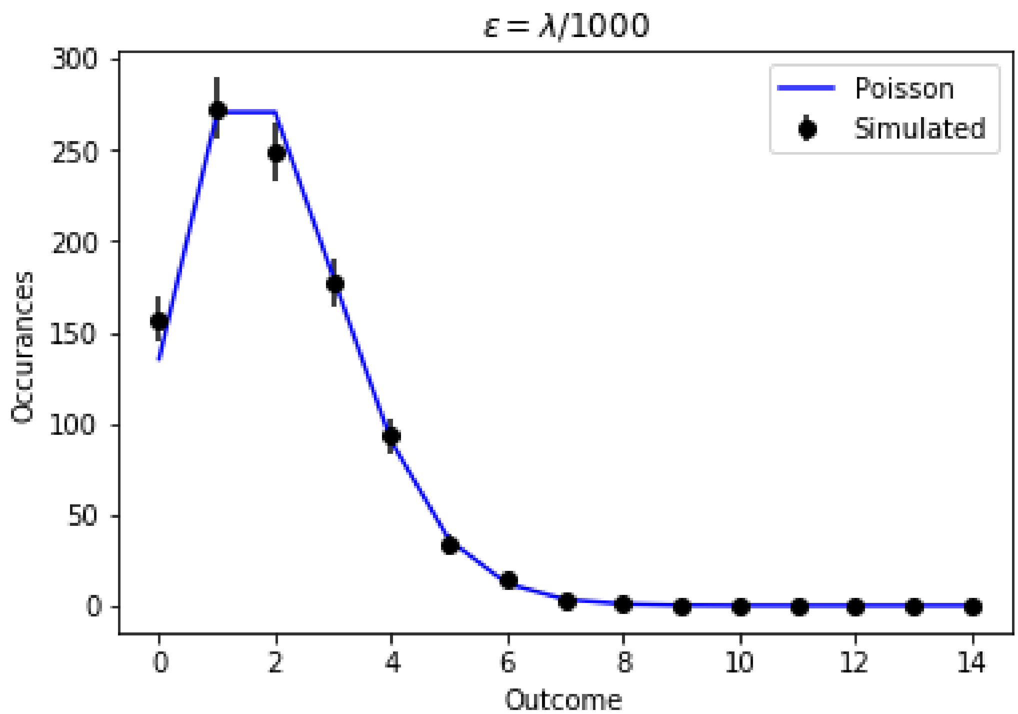
[59]:
```
#7.5 Comparison with Binomial PDF

from scipy.stats import binom

# MCBD = input("Compare Monte Carlo with The Binomial Distribution? 'yes' or␣
↪'no': ")
# yes
# if MBCD == string

errs = counts**0.5
plt.errorbar(edges[:-1], counts, yerr = errs, fmt = "ko", label = "Simulated")

plt.xlabel("Outcome")
plt.ylabel("Occurances")
xpred = edges[:-1]
ypred = nexp * binom.pmf(xpred, ntry, eps)
plt.plot(xpred, ypred, "b-", label = "Binomial")
plt.legend()
plt.title("$\epsilon = 0.25$")
```

[59]:
```
Text(0.5, 1.0, '$\\epsilon = 0.25$')
```



[60]:
```
#7.6 The Poisson Limit
#note

from scipy.stats import poisson

nexp1 = 1000
ntry1 = 1000
lamb1 = 2
eps1 = lamb1/ntry1
v1 = th_bi(nexp1, ntry1, eps1)
counts, edges = np.histogram(v1, bins = 15, range = (0,15))

errs = counts**0.5
plt.errorbar(edges[:-1], counts, yerr = errs, fmt = "ko", label = "Simulated")

plt.xlabel("Outcome")
plt.ylabel("Occurances")
xpred = edges[:-1]
ypred = nexp * poisson.pmf(xpred, lamb1)
plt.plot(xpred, ypred, "b-", label = "Poisson")
plt.legend()
plt.title("$\epsilon = /1000$")
```

[60]: Text(0.5, 1.0, '$\\epsilon = /1000$')

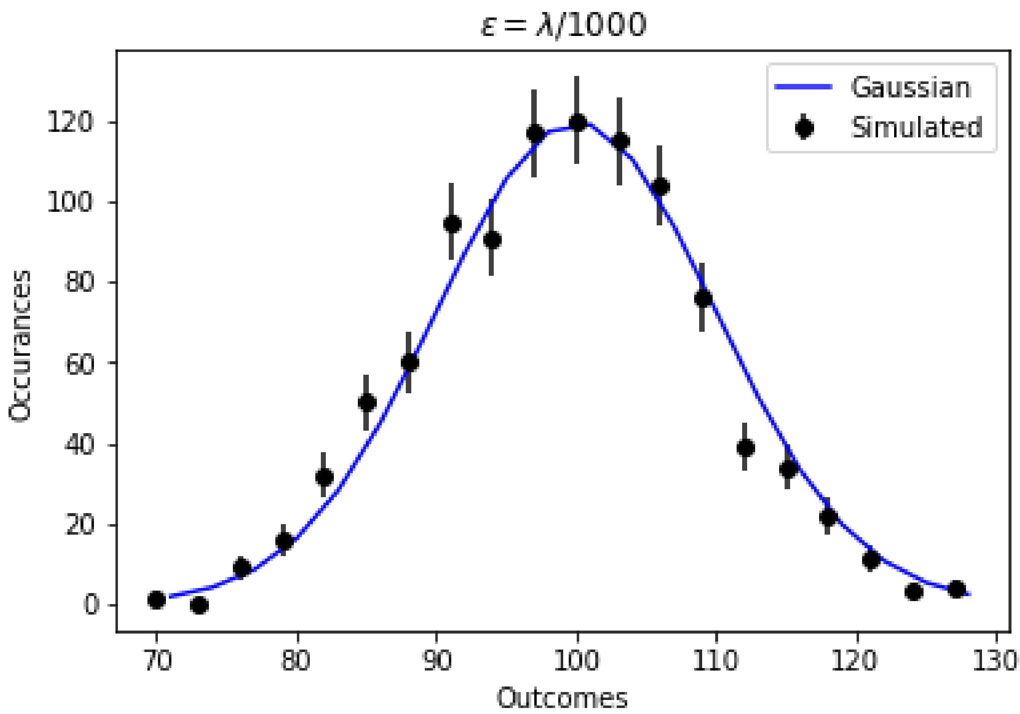


[63]:
```
#7.7 The Gaussian Limit

from scipy.stats import norm


lamb2 = 100
eps2 = lamb2/ntry1
v2 = th_bi(nexp1, ntry1, eps2)
counts, edges = np.histogram(v2, bins = 20, range = (70,130))
w = edges[1] - edges[0]

errs = counts**0.5
plt.errorbar(edges[:-1], counts, yerr = errs, fmt = "ko", label = "Simulated")
print(counts)
cbins = (edges[:-1] + edges[1:] - 1)/2
plt.xlabel("Outcomes")
plt.ylabel("Occurances")
xpred = cbins
ypred = nexp * w * norm.pdf(xpred, loc = lamb2, scale = lamb2**0.5)
plt.plot(xpred, ypred, "b-", label = "Gaussian")
plt.legend()
```

```
plt.title("$\epsilon = /1000$")
```

```
[  1   0   9  16  32  50  60  95  91 117 120 115 104  76  39  34  22  11
   3   4]
```

[63]:
```
Text(0.5, 1.0, '$\\epsilon = /1000$')
```



[ ]: