1. Problem Sheet SDA

Joanna Radack radack@uni-potsdam.de (mailto:radack@uni-potsdam.de)

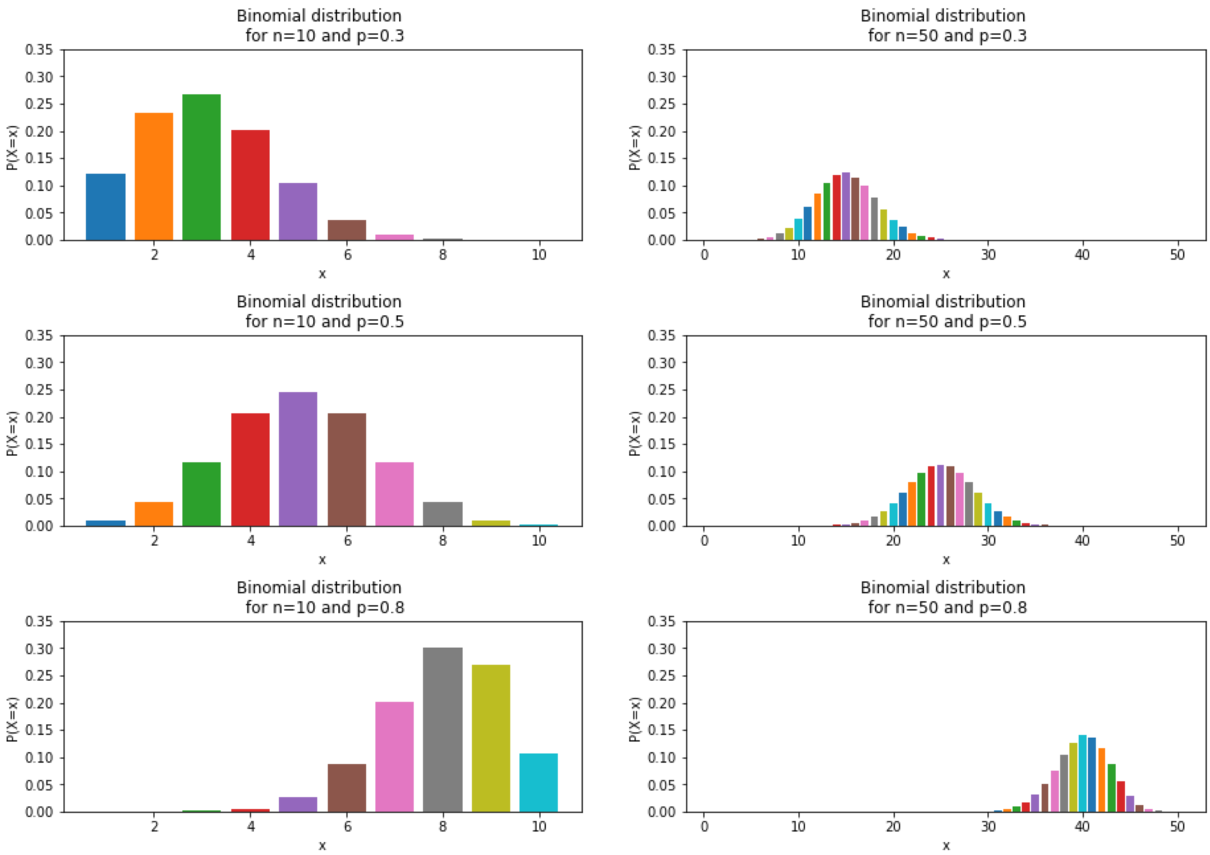
In [2]:
```
import math
import matplotlib.pyplot as plt
import scipy.special
```

In [3]:
```
p = [0.3,0.5,0.8] #list containing probabilities
n = [10,50] #list containing number of repetitions in an experiment
l = [0.3,2,6] #list of lambdas for the poisson distribution

plt.rcParams['figure.figsize'] = [15, 10] #increasing size of plots
```

Exercise 3.1

In [4]:
```
plt.figure(1)
plotid=321
for j in range(len(p)):
    for i in range(len(n)):
        #To create my necessary subplots without having to repeat the same code,
        #I created 'for loops' that first specifies our p, then runs through
        #the different numbers of repetition, and then repeats for our remaining probabilities
        plt.subplot(plotid)
        plt.xlabel('x')
        plt.ylabel('P(X=x)')
        plt.title('Binomial distribution \n for n={} and p={}'.format(n[i],p[j]))
        plt.ylim([0, 0.35])
        # Here I am just formatting and labelling my figure.
        for k in range(1,n[i]+1):
            biko = scipy.special.binom(n[i],k) #binomial coefficient
            plt.bar(k, biko * p[j]**k * (1-p[j])**(n[i]-k)) #formula for the binomial distribution
            #For each k, or our number of successes, the bernoulli probability is calculated. The 'for loop'
            #starts with a set n and p and runs through our k variable up until the maximum number of successes (=
        plotid+=1
plt.subplots_adjust(left=0.1, #just some formatting to make the plots look nice
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.5)
```



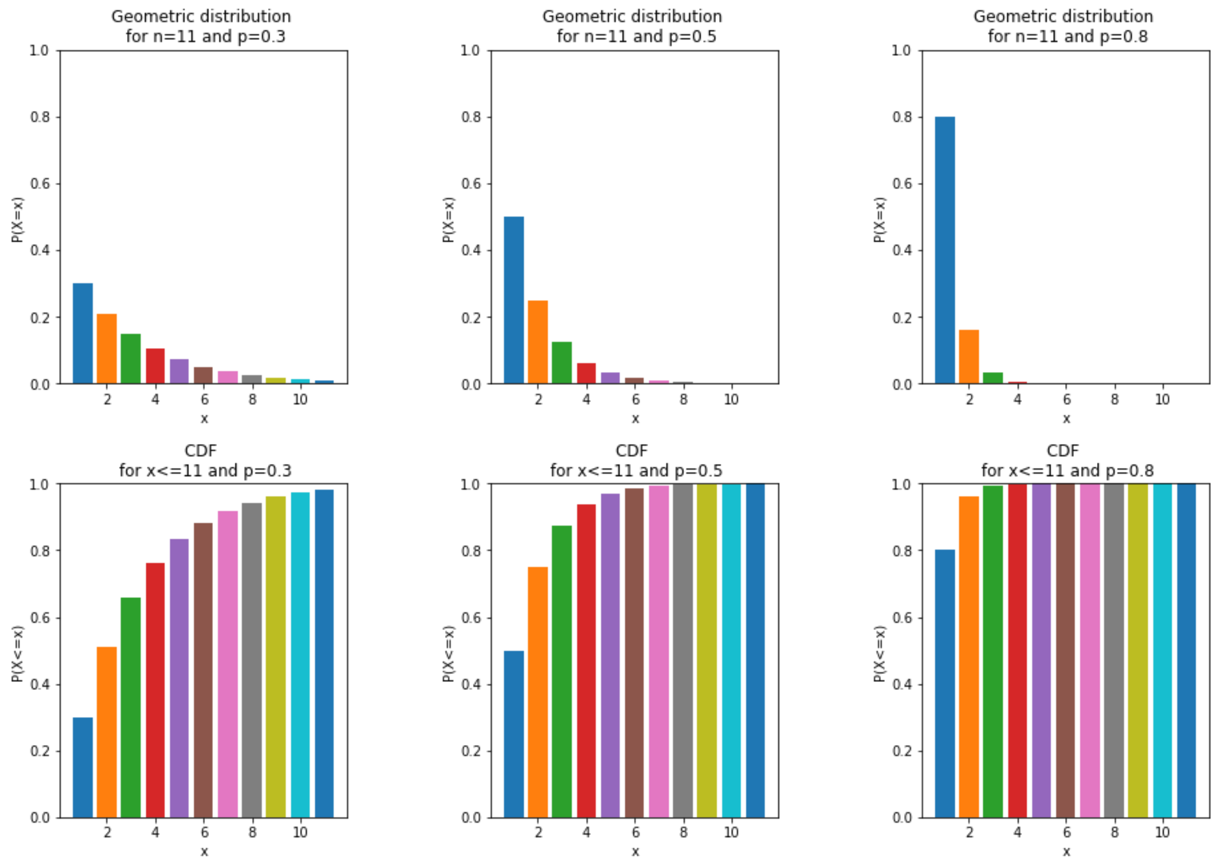
In all cases, the binomial distribution clearly shows the expected value (n*p) having the highest probability. When comparing the distributions of n=10 and n=50 though, the probability of a specific value is decreased in the distributions with n=50. But the plots also clearly show that the distribution itself is more evenly spread around the expected value, approaching the Gauss'sche bell curve.

Exercise 3.2

In [92]:

```
plt.figure(2)
plotid=231
for a in range(len(p)): #'a' once again runs through our list of probabilities
    plt.subplot(plotid)
    plt.xlabel('x')
    plt.ylabel('P(X=x)')
    plt.title('Geometric distribution \n for n=11 and p={}'.format(p[a]))
    plt.ylim([0, 1])
    for b in range(1,12):
        #'b', representing our number of repetitions until our first success, can take the values 1-11,
        #as we have our x (number of repetitions) set at 11 and we need at least 1 try to get a success
        plt.bar(b, p[a]*(1-p[a])**(b-1))
        #this is the formula for the geometric distribution, calculating the probability of a success after 'b' re
    plotid+=1

for m in range(len(p)):
    plt.subplot(plotid)
    plt.xlabel('x')
    plt.ylabel('P(X<=x)')
    plt.ylim([0, 1])
    plt.title('CDF \n for x<=11 and p={}'.format(p[m]))
    cdf = p[m]
    #our cumulative distribution function starts at P(X<=1), therefore the first value is the geometric distributi
    plt.bar(1, cdf)
    for o in range(2,12):
        cdf = cdf + p[m]*(1-p[m])**(o-1)
        #the rest of the cumulative distributions are continuously added to the previous cdf until our total numbe
        plt.bar(o, cdf)
    plotid+=1
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.5,
                    hspace=0.3)
```

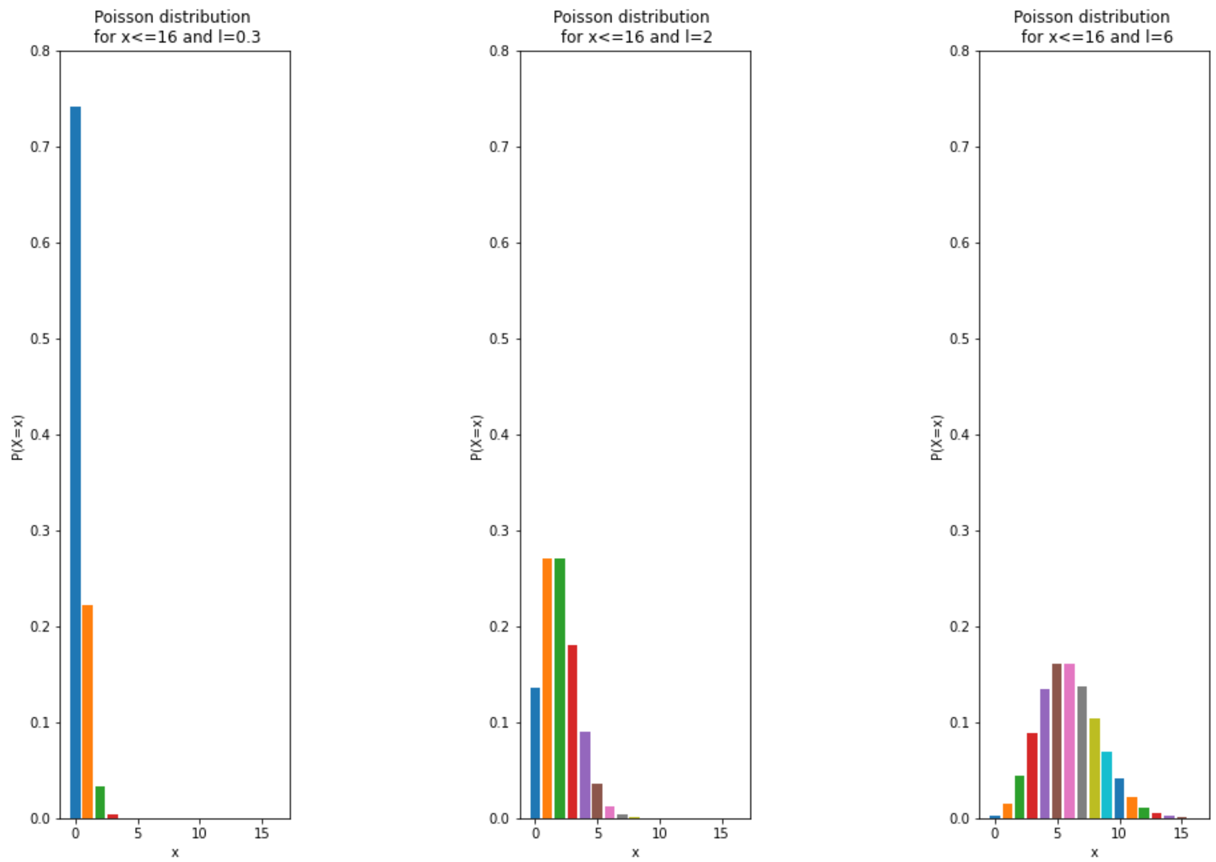


In these plots it can be seen that for the same number of repetitions, the higher the given probability of an event occuring is, the more unbalanced the geometric distribution is. But for all probabilities plotted, the probability that the first success occurs at a later repetition decreases continously (it is most likely that the first success already occurs after the first repetition). The cdfs show this fact as well, by the quick rise to almost 100% certainty of getting a success in all examples. The cdf of p=0.8 already reaches almost 100% certainty after at most 3 repetitions.

Exercise 3.3

In [93]:

```
plt.figure(3)
plotid=131
for k in range(len(l)):
    plt.subplot(plotid)
    plt.xlabel('x')
    plt.ylabel('P(X=x)')
    plt.title('Poisson distribution \n for x<=16 and l={}'.format(l[k]))
    plt.ylim([0, 0.8])
    for g in range(17): #This time our x is set to 16
        plt.bar(g, math.e**(-l[k])*((l[k]**g)/math.factorial(g))) #to calculate the poisson distribution one needs
        #a lambda is set, then the probability of g successes is calculated
    plotid+=1
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=1,
                    hspace=1)
```



As shown by these plots, a small lambda such as 0.3 results in a very unbalanced probability distribution, where the probability of not having any successes is very high. Whereas the large lamba 6 looks similar to a binomial distribution, where the probabilities are spread more evenly around the expected value.