1.3 Sampling

The `math`, `numpy` and `scipy` libraries contain functions to generate random variates. That is realization of random samples. For example the following command generates a random variates of size 10 of a probability distribution $Ber(1/2)$

```
P = stats.bernoulli(0.5)
s = P.rvs(10)
```

Exercise: - Execute the code given above. - Print the value of `s`. What kind of Python object is it? - Do you know the mathematical definition of a *random sample*? and a *random variates*?

```
In [106]: P = stats.bernoulli(0.5)
          s = P.rvs(10)

In [107]: print(s)
          print(type(s))

[1 1 0 0 1 1 1 1 1 1]
<class 'numpy.ndarray'>


In [108]: # a random sample is a sequence of independent and
          # identically distributed (iid) random variables
          #  X_1, X_2, X_3, ...
          # a random variates is a *realization* if this sequence
          # that is X_1(omega), X_2(omega), X_3(omega), ...
          # for some omega in the universe

In [ ]:

In [ ]:
```

WARNING: In many situation there is an abuse of language between a random sample and a random variates. Computers only deal with random variates.

Given a finite set of real numbers $A = \{x_1, x_2, \ldots, x_n\}$ we associate the following (discrete) probability measure

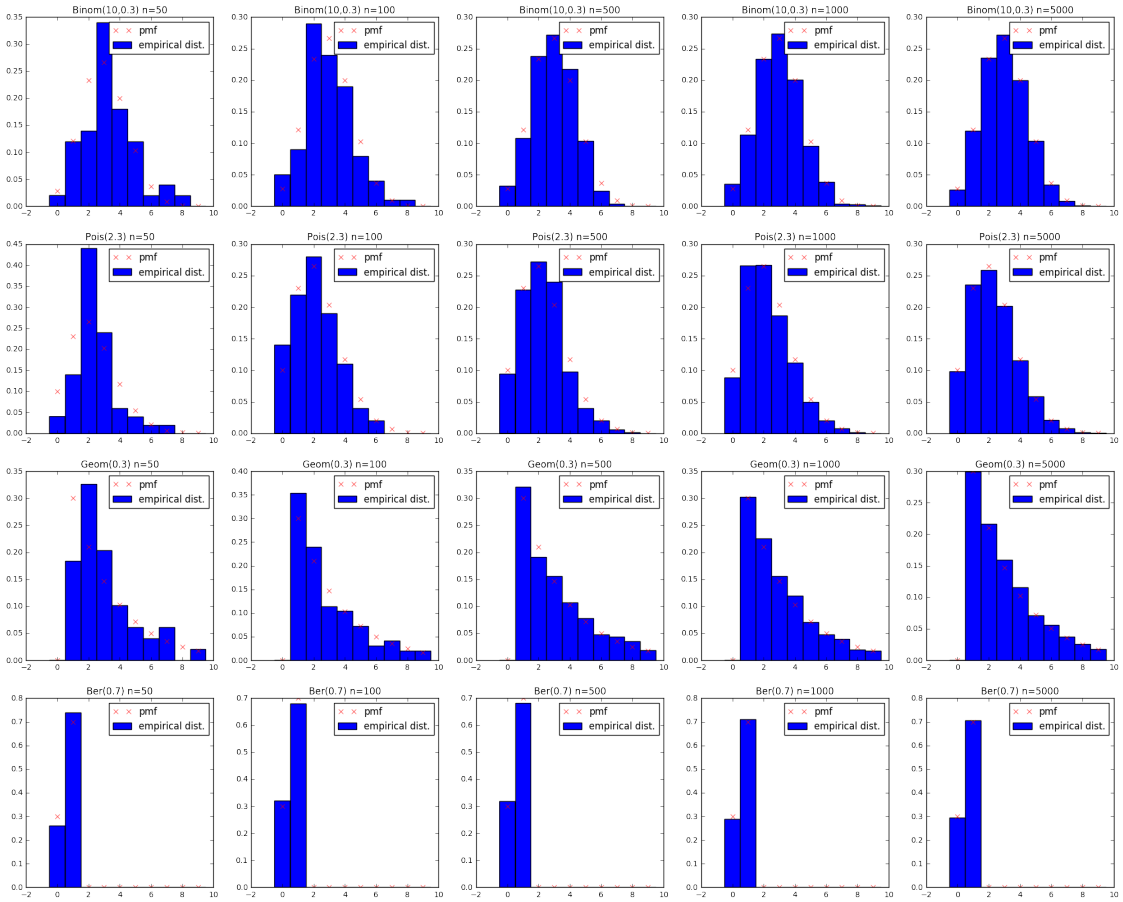
$$\mu_A = \frac{1}{\#A} \sum_{a \in A} \delta_a = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

This measure is called the *empirical distribution*.

Exercise: For each of the probability measures `P1`, `P2`, `P3`, `P4`, `U`, `N` do the following: - Construct a random sample of size 50. - Using `plt.hist` make a graphics of the empirical distribution of the sample. - Add the graph of the probability mass function (discrete case) or the density function (continuous case) to the graphics. - Repeat these operations with samples of size 100, 500, 1000, 5000.

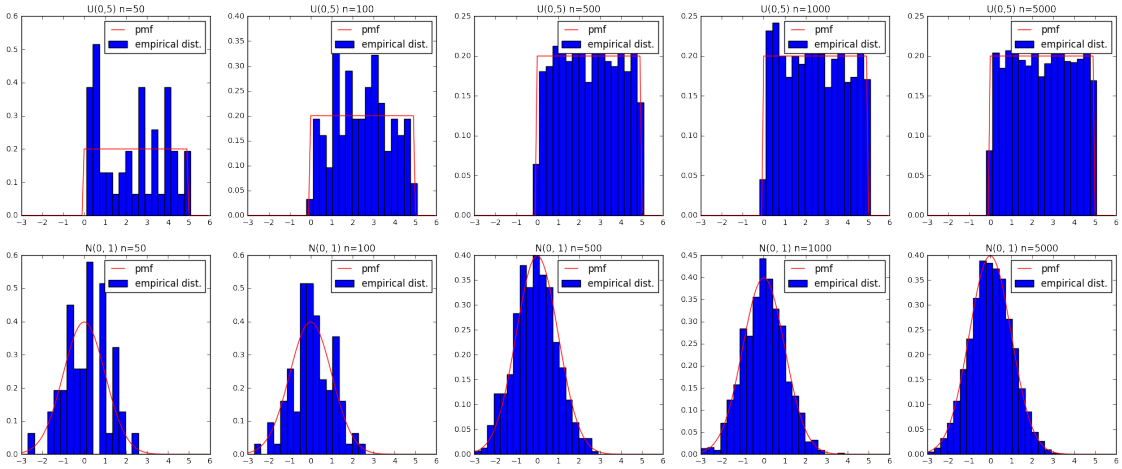
By analyzing the graphics, can you say what does the empirical distribution as the size of the sample gets bigger? Do you know how to prove it?

In [153]:
```
# the case of discrete probabilities
# each row corresponds to a fixed probability distribution
# each column corresponds to a fixed sample size
i = 1
for P, name in discrete_probas:
    for n in [50, 100, 500, 1000, 5000]:
        ax = plt.subplot(4, 5, i)
        s = P.rvs(n)
        ax.hist(s, bins=np.arange(-0.5,10.5,1), normed=True, label="empir
        x = np.arange(0,10,1)
        ax.plot(x, P.pmf(x), 'xr', label='pmf')
        ax.legend()
        ax.set_title(name + " n=%d" % n)
        i += 1
fig = plt.gcf()
fig.set_figwidth(25)
fig.set_figheight(20)
plt.legend()
plt.show()
```



In [ ]:

In [154]:
```
# the case of discrete probabilities
# each row corresponds to a fixed probability distribution
# each column corresponds to a fixed sample size
i = 1
for P, name in continuous_probas:
    for n in [50, 100, 500, 1000, 5000]:
        ax = plt.subplot(2, 5, i)
        s = P.rvs(n)
        ax.hist(s, bins=np.linspace(-3, 6, 30), normed=True, label="empir
        x = np.arange(-3, 6, 0.1)
        ax.plot(x, P.pdf(x), 'r', label='pmf')
        ax.legend()
        ax.set_title(name + " n=%d" % n)
        i += 1
fig = plt.gcf()
fig.set_figwidth(25)
fig.set_figheight(10)
plt.legend()
plt.show()
```



In [ ]:

1.4 Toss a coin

We now turn to coin tossing, that is a sequence of independent random variables with probability distribution $Ber(1/2)$. Random variables were introduced in the Lecture 2 of J. Nzabanita.

Exercise: - What does the following code

```
P = stats.bernoulli(0.5)
t = [np.count_nonzero(P.rvs(10)) for i in range(20)]
```