The Central Limit Theorem in Action

Goal

You are studying the central limit theorem in the statistics boot camp or have seen it before. Our goal is to observe the central limit theorem in action whereby the means of samples drawn from a uniform random variable follow a normal distribution.

Discussion

Statistical inference is all about estimating statistics about a population from a sample or samples. The *central limit theorem* (CLT) is a critical piece of the machinery that allows us to infer things about a population.

In a nutshell, the CLT says that if we look at the sample means for a bunch of samples drawn from some distribution with mean μ and variance σ^2, the means will bounce around the true population mean. That means that the sample mean is itself a random variable and the CLT tells us the distribution is $N(\mu, \sigma^2/n)$ for sample size n. Things to note:

1. X's distribution doesn't really matter (that's close enough to the truth for our purposes)—the distribution of its sample mean is normal.
2. The variance of the sample mean (not the sample variance), σ^2/n, gets tighter as we increase the sample size n.
3. More trials (number of X samples we conjure up) improves the resolution of the sample-mean normal distribution but doesn't change the variance. In other words, more samples help our visualization of the sample mean distribution but it does not change our estimate of the sample mean.

We are going to observe the sample mean distribution for a uniform random variable in $U(a = 0, b = 1)$ with $\mu = \frac{a+b}{2} = 0.5$ and $\sigma^2 = \frac{(b-a)^2}{12} = \frac{1}{12}$. The histogram should therefore show normal distribution $N(0.5, (1/12n))$ for sample size n.

Steps

The complete code is in `notes/code/clt.py` but here are the key steps.

1. Get TRIALS=500 samples X of size $n = 4 = LEN(X)$ from the uniform distribution $U(0, 1)$ using our `runif01()` function or Python's `random.random()`. Compute the mean of each X vector and add it to the end of a different array `X_`.

2. Plot the histogram of `X_` with `bins=60, normed=1`.

3. Let's add the theoretical normal distribution on top. To do that we need the appropriate parameters of $N(\mu, \sigma^2/n)$. The mean μ of uniform samples should be midway between a and b from $U(a, b)$. In our case, that's 0.5 since we are doing $U(0, 1)$. The variance of the uniform distribution is $(b-a)^2/12$ and we need the variance divided by sample size n. We can define a function that returns the variance of uniform distribution $U(a, b)$:

```
def unifvar(a, b):
    return ((b - a) ** 2) / 12.0
```

4. To get the theoretical distribution, let’s define it ourselves from

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 \big/ 2\sigma^2}$$

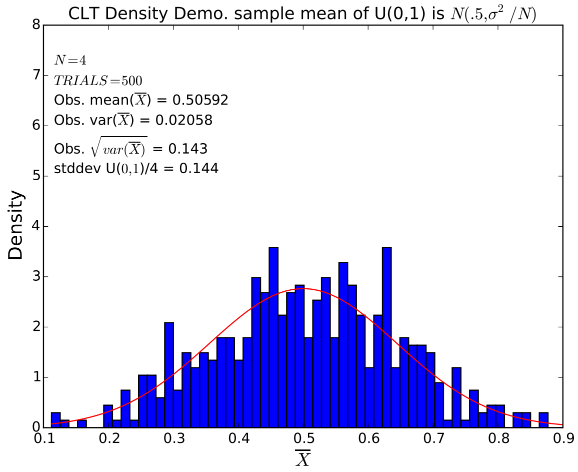
```
def normpdf(x, mu, sigma): # sigma is the standard deviation, sigma^2 is the variance
    """
    Accept either a floating-point number or a numpy ndarray, such as what you get
    from arange().  You do not need a loop in the code does not change here
    because 2 * ndarray is another ndarray automatically. In this respect,
    numpy is very convenient and behaves like R.
    """
    u = - (x - mu)**2 / (abs(sigma)**2 * 2.0)
    y = (1.0 / (math.sqrt(2 * math.pi) * abs(sigma))) * math.e ** u
    return y
```

5. Then, plot the theoretical normal distribution on top of the histogram. (Also set the axes so that we can use the same range throughout the next series of tests to see how the distribution changes.) Note that the usual normal density function provided above expects the **standard deviation not the variance** and so we need to pass `normpdf()` the square root of the expected variance.

6. Now, display some important parameters in the graph using `text()`. You will need to do the `ax = fig.add_subplot(111)` thing early in your script. The text in between the $ symbols is latex and lets us display nice math symbols (e.g., the title), although I’m not doing much with it here.

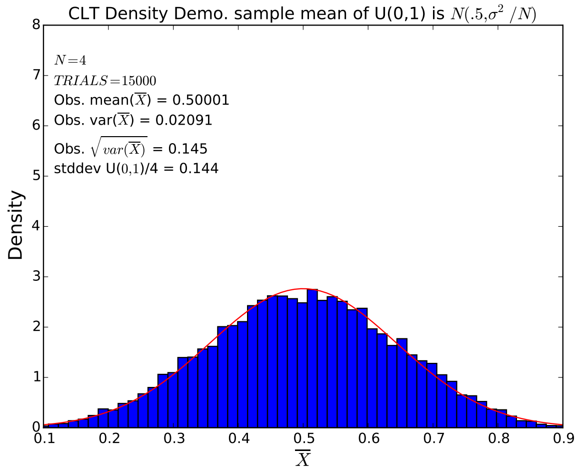
```
plt.text(.02, .9, ’$N = %d$’ % N, transform=ax.transAxes)
plt.text(.02, .85, ’$TRIALS = %d$’ % TRIALS, transform=ax.transAxes)
plt.text(.02, .8, ’Obs. mean($\\overline{X}$) = %5.5f’ % np.mean(X_), transform=ax.transAxes)
plt.text(.02, .75, ’Obs. var($\\overline{X}$) = %5.5f’ % np.var(X_), transform=ax.transAxes)
plt.text(.02, .68, ’Obs. $\sqrt{var(\\overline{X})}$ = %3.3f’ % np.sqrt(np.var(X_)), transform=ax.transAxe
plt.text(.02, .63, ’stddev U($0,1$)/%d = %3.3f’ % (N, np.sqrt(sample_mean_var)), transform=ax.transAxes)

plt.title("CLT Density Demo. sample mean of U(0,1) is $N(.5, \sigma^2/N)$")
plt.xlabel("$\\overline{X}$", fontsize=16)
plt.ylabel("Density", fontsize=16)
```

7. Run it. The resulting graph should look like this

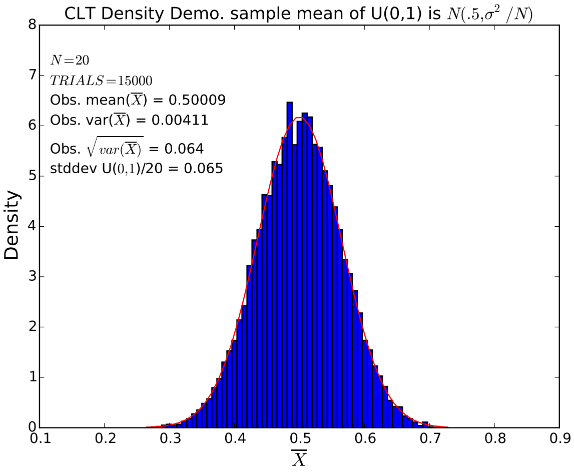


Notice how the mean is close to the expected 0.5 and that the observed stddev of the sample means is close to the theoretical stddev.

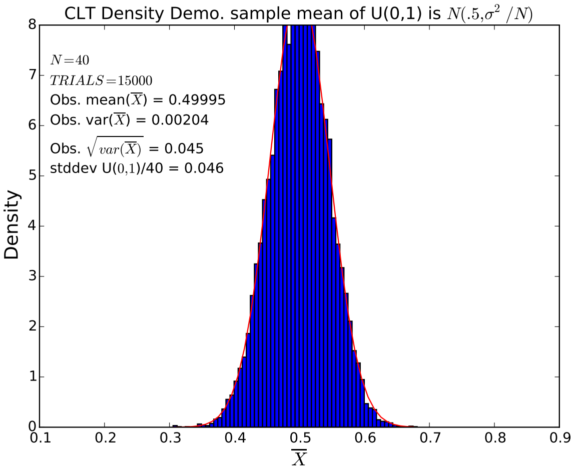
8. Increasing the number of trials to 15,000 shows much higher resolution but does not change the variance/tightness of the distribution at all:



9. Now, if we increase the sample size to $N = 20$, we get a much tighter variance on the mean of $\overline{X}$. Run it:



10. Increasing to 40 we get:


CLT Density Demo. sample mean of U(0,1) is $N(.5,\sigma^2/N)$
$N=40$
$TRIALS=15000$
Obs. mean($\overline{X}$) = 0.49995
Obs. var($\overline{X}$) = 0.00204
Obs. $\sqrt{var(\overline{X})}$ = 0.045
stddev U(0,1)/40 = 0.046
Density
$\overline{X}$