The Central Limit Theorem

The purpose of this exercise is two-fold. First, to further introduce some of the tools and commands in **R**. Second, we will show one consequence of the central limit theorem, namely that sums of independent random variables tend towards a normal. Specifically, we can generate draws from a highly non-normal distribution, but the sample mean of such draws tends toward a normal distribution.

Step 1: Let's generate some Gamma random variables.

The gamma distribution generates a value on the open interval from zero to infinity. It is defined by two (positive) parameters, a shape and a scale parameter. In **R**, the gamma is specified by **`gamma(shape, scale)`**. Specifically, if shape = α and scale = β, the probability density is given by

$$f(x) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}$$

for $x > 0$, $\alpha > 0$ and $\beta > 0$. $\Gamma()$ is the gamma function. The mean and variance are

$$E(X) = \alpha * \beta, \quad \text{and} \quad Var(X) = \alpha * \beta^2$$

Both the exponental ($\alpha = 1$) and χ^2 distributions ($\alpha = df/2$, $\beta = 2$) are special cases of the gamma.

• **Task 1:** Use the **`curve`** command to explore how different shape and scale values alter the distribution. Use

```
curve(dgamma(x,shape, scale))
```

where you will supply different numeric values for shape and scale. Recall **R** syntax, where if **`DIST`** is the distribution, then **`dDIST`** is the probability density function. If you wish to restrict the plot between (say) 0 and 30, you would use

```
curve(dgamma(x,shape, scale),0,30)
```

Our first **R** notes in the class further tells you how to do cool things like plot multiple curves, each with a different color, on the sample plot. Try plotting two or three gamma distributions, holding one parameter constant, but changing the other (i.e., keep the shape value the same, but change the scale value). This sort of playing around is a good way to get a feel for the effects of parameters on the shape of the distribution.

• **Task 2:** After playing around with various shape and scale parameters, pick a set of particular values that you feel generates a very non-normal looking distribution (i.e., sure does not look like a symmetric bell-shaped curve). A very useful

R command allows you to generate **n** draws from any specified distribution, it is **rDIST(n, parameters)**. For a gamma, this syntax implies the command

```
rgamma(n,shape,scale)
```

Of course, when you type this command, you will supply numerical values for the random of draws **n** and the two gamma distribution parameters. For example, to compute 40 random draws from a gamma with shape = 1 and scale = 10, we would use

```
rgamma(40,1,10)
```

When you type in this command, **R** happy spits back to your screen all forty (or **n**) values. Hence, don't try this using n = 5,000 unless you want to fill your screen (go ahead and try it if you wish)

A useful way around this is to simply place all of the generated values into a vector (let's call it **c**) that **R** keep track of. Recalling the **R** syntax of **<-** means assignment, we use

```
c <- rgamma(n,shape,scale)
```

When you type in this command, **R** returns nothing to the screen, but instead stores all of the values into **c** (or whatever name you choose). If case you are worried, type **c** at the **R** prompt, and you will be rewarded with seeing all of the values.

Now generate 60 draws from your favorite gamma and store these to **c**. **R** has easy command to summarize this data. For example

```
mean(c)
```

returns the mean of the elements in **c**, while

```
var(c)
```

returns the variance, and

```
sd(c)
```

the standard deviation. Now, to see that your draws are highly-normal, you can plot out a histogram of the value using

```
hist(c)
```

You can also use several tests of normality. One favorite is the **Wilks-Shapiro** test, which jointly tests for zero skewness and (scaled) kurtosis = 3. Deviation from either shows departures from normality. To apply this test to your vector of values, use

```
shapiro.test(c)
```

R returns the p value for the fit to a normal. You can also look at the **quantile versus quantile** (or **q-q**) **plot**. This is a plot that compares the cumulative distribution of your data with the expected cumulative distribution of values from some specified distribution. Devitions from a straight line indicate departure from the specified distribution. In **R**,

```
qqnorm(c)
```
plots the values in **c** versus those expected form a normal. You can add a straight line with the command **qqline(c)** or you can apply both at once using a semi-colon,

```
qqnorm(c); qqline(c)
```
One final test for goodness of fit is a **Kolmogorov-Smirnov test**, which is a non-parametric test based on comparing the absolute difference between the cumulative distribution functions of two candidates. It has lower power than Wilks-Shapiro (not surprising, as WS is specfically designed for testing against a normal). You want to compare your distribution to a normal with the same mean and variance (measured by the deviation) as your sample. The **R** command for this is

```
ks.test(c, "pnorm", mean = mean(c), sd = sd(c))
```
The syntax here is that **ks.test** calls up the Kolmogorov-Smirnov test, using a normal distribution (**"pnorm"**) with the same mean and standard deviation as in our sample (**c**). You should notice that the p value is much larger under KS than for the WS test on the same data (showing that KS has less power)

• **Task 3:** Now let's generate 400 sets of draws from your gamma. We will compute the mean for each draw, and store these 400 sample means in the vector **samp** (or whatever you like). First, we need to tell **R** to reserve a vector of length 400, whose values are numeric

```
samp <- numeric(400)
```
Now recall from above that

```
mean(rgamma(60,shape,scale))
```
will return the mean of 60 random draws from your gamma (remember that you have to supply numerical values for the scale and shape parameters). To generate 400 such means, we will use the **for** loop syntax in **R**, specifically,

```
for(i in 1:400) samp[i] <- mean(rgamma(60,shape,scale))
```
With this command **R** will, for values of i from 1 to 100, assign the i-th element in the vector **samp** the mean from that i-th set of random draws.

Now, using the commands above, plot the histogram of this vector of means. Looks a lot closer to a normal than the individual values for any particular draw. Now test this, using the WS and KS tests, as well as examining the fit graphically using qq-plots.

Now, repeat task three using both a smaller number of draws (sample size) for each set (say 20) and several larger sample sizes (say 100, 200, and 500). What happens to the goodness of fit of the distribution of the sample mean to a normal as the sample size (number of draws in each set) increases?