Simulation of a Convergence to the Normal Distribution

Jim Callahan

September 17, 2015

Overview

It is a useful concept in probability and statistics that even if a probablity distribution is not normal (does not follow the "bell curve") that if you calculate means (averages) of small subsamples from the distribution the distribution of the means will be normal, as long as the underlying distribution has a finite variance (for example, not a power law distribution) and meets a few less well known mathematical properties.

In this project we demonstrate how averages of an asymetric distribution (the possion distribution) converge to a symetric normal distribution.

The convergence to normal by an average is an application of the Central Limit Theorem (CLT). "For our purposes, the CLT states that the distribution of averages of iid [] variables, properly normalized, becomes that of a standard normal as the sample size increases" slide from Dr. Brian Caffo's *"Statistical Inference"* class slide 8/20 "A Trip to Asymptopia" in Asymptopia.pdf

Simulations

This project uses the programming language **R** to simulate draws from an **exponential distribution**.

We first look graph and simulation of the **exponential distribution** and then compute and graph a mean (average) of sets 40 observations each from the exponential distribution and finally compare the distribution of the means with the normal distribution to see if the process of applying the means has "shape shifted" the distribution from exponential to normal.

The draws from the exponential distribution are simulated in **R** with a random number function. Random number functions typically return values either a normal or a uniform distribution. This simulation depends on a family of random number functions in **R** that are modified to return random values from specific probablity distibutions other than the normal distribution.

Specifically, **rexp()** is in **R** a specialized random number function that returns values from the **exponential distribution**:

```
x <- rexp(n, lambda)
where:
n     = the number of draws
lamda = the rate parmater
        (for purposes of this project lambda = .2)
x     = the values of x returned by the draw.
```

In **theoretical** terms, the **exponential distribution** has the property that the expected value of the **mean** and **standard deviation** (the standard deviation is the square root of the variance) are both equal to the inverse of **lambda (1/lambda)**.

```
Exponential Distribution
mean             = 1/lambda (in this project 1/.2 = 5)
std deviation    = 1/lambda (in this project 1/.2 = 5)
std deviation    = squareroot(variance)
variance         = square of the standard deviation (sd^2)
```

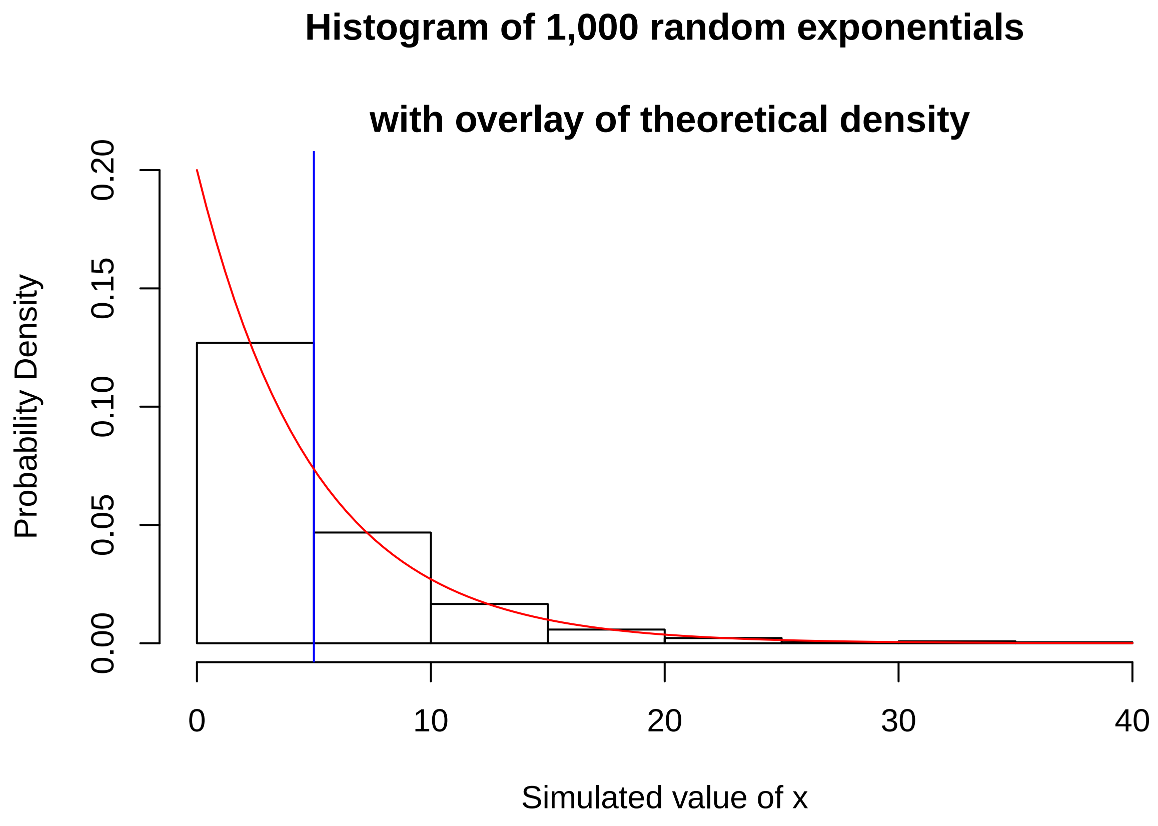
We begin with graph and simulation of the **exponential distribution** itself; after that we will compute and graph a mean (average) of sets 40 observations each from 1,000 simulations.

```r
# set the random number seed so the results are reproducible.
set.seed(1234)
# Simulate 1000 random exponentials
x <- rexp(1000, rate = 0.2)

# Make the Y-axis start at zero and go to the maximum value
# in the range() function.
ylim <- range(0, .20)

# Plot the empirical results of the simulation
# Use density instead of frequency,
# so the theoretical curve will fit.
hist(x, freq = F, ylim = ylim,
     main = "Histogram of 1,000 random exponentials
             \n with overlay of theoretical density",
     xlab = "Simulated value of x",
     ylab = "Probability Density"
     )

# Overlay a plot a theoretical exponential density curve
abline(v = 5, col="Blue")
curve(dexp(x, rate = 0.2), add=T, col="Red")
```



A simulation of 1,000 values of the exponential distribution with a histagram of the raw (untransformed) simulated values of x without taking an average. For comparison we overlaid (the smooth red curve) the theoretical exponential distribution. The vertical blue line shows the expected mean of 5 when lambda = .2. ### Sample Mean versus Theoretical Mean We know the theoretical expected mean should approach 5 (when lambda = .2); what do we get when we calculate the mean of the specific values of x we drew with the simulation?

```
mean(x)
```

```
## [1] 5.003067
```

Sample Variance versus Theoretical Variance

We also know the theoretical expected standard deviation should approach 5 (when lambda = .2). Because the standard deviation is defined as the square root of the variance; the variance should be the square of the standard deviation; which in this case with a standard deviation of 5 the standard devistion squared or variance should approach 25. When we calculate the standard deviation and variance of the specific values of x we drew with the simulation, we get:

```
sd(x)        # standard deviation
```

```
## [1] 5.056718
```

```
(sd(x))^2    # standard deviaiton squared
```

```
## [1] 25.5704
```

```
var(x)       # variance
```

```
## [1] 25.5704
```

Include figures with titles. In the figures, highlight the means you are comparing. Include text that explains the figures and what is shown on them, and provides appropriate numbers.

Include figures (output from R) with titles. Highlight the variances you are comparing. Include text that explains your understanding of the differences of the variances.

Distribution

Via figures and text, explain how one can tell the distribution is approximately normal.

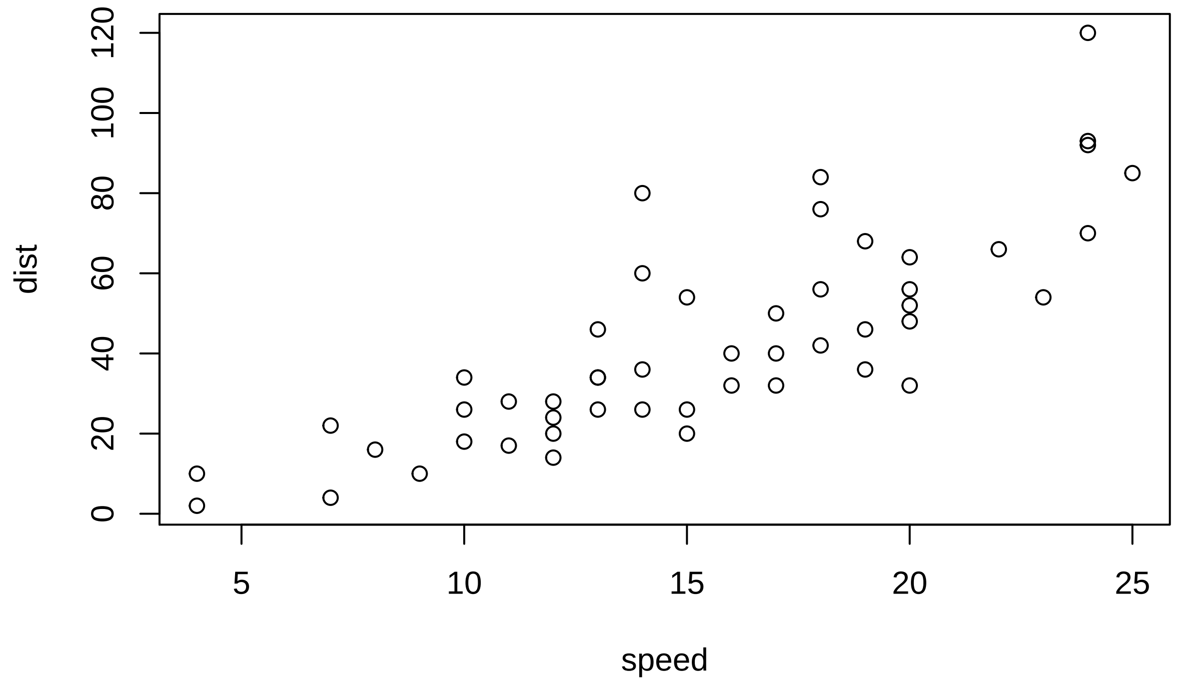
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.