Statistical Inference Course Project

Yevgen Yampolskiy

01/21/2015

In this project you will investigate the exponential distribution in R and compare it with the Central Limit Theorem. The exponential distribution can be simulated in R with `rexp(n, lambda)` where lambda is the rate parameter. The mean of exponential distribution is `1/lambda` and the standard deviation is also `1/lambda`. Set `lambda = 0.2` for all of the simulations. You will investigate the distribution of averages of 40 exponentials.

I'm using a huge number of simulations (1,000,000) to have better support my final point that average of 40 exponentials is very far from being normal. For the rest of discussion smaller values of simulations (100,000) would be a bit more attractive. Note that non-normality is easily seen even with 100,000 simulations.

Draw samples from exponential distribution

```
set.seed(12345)
number.of.simulations <- 1000000
number.of.variables <- 40
lambda <- 0.2
draw.data <- as.data.frame(matrix(
  rexp(number.of.variables * number.of.simulations, lambda),
  nrow = number.of.simulations,
  ncol = number.of.variables))
```

Create sample of averages

So far we just draw a bunch of tuples of 40 random values. The goal is to estimate distribution of averages, so we need to compute a sample of averages (of 40 exponentials).

```
data <- rowMeans(draw.data)
names(data) <- 'sample.means'
```

Sample mean computation

We have a population of means of 40 draws from exponential distribution. We took a sample of `number.of.simulations` from this distributions. Now we use sample mean to estimate the true mean of this population (using `mean` function).

```
sample.mean <- mean(data)
theoretical.mean <- 1.0 / lambda
print(sprintf("Sample mean: %f, True mean: %f, Error: %f",
              sample.mean, theoretical.mean, sample.mean - theoretical.mean))
```

```
## [1] "Sample mean: 5.000218, True mean: 5.000000, Error: 0.000218"
```

Sample variance computation

I will compute sample variance using sample mean because that is the only thing we really have in practice. Since sample mean is derived from the same sample we reduce the number of degrees of freedom by one: `number.of.simulations - 1`.

As of theoretical diviation, for the mean for a sample of size `n` is `sigma/sqrt(n)`.

```
sample.variance <- sum((data - sample.mean)^2) / (number.of.simulations - 1)
sample.sigma = sqrt(sample.variance)
theoretical.sigma <- (1.0 / lambda) / sqrt(number.of.variables)
print(sprintf("Sample diviation: %f, True diviation: %f, Error: %f",
              sample.sigma, theoretical.sigma, sample.sigma - theoretical.sigma))
```

```
## [1] "Sample diviation: 0.790723, True diviation: 0.790569, Error: 0.000153"
```

Confidence interval for the mean estimate

Population mean was estimated using large number of observations (`number.of.simulations`), so we can use quantiles of a normal distribution to estimate 95% confidence interval:

```
print(sample.mean + c(-1, 1) * qnorm(0.975) * sample.sigma / sqrt(number.of.simulations))
```

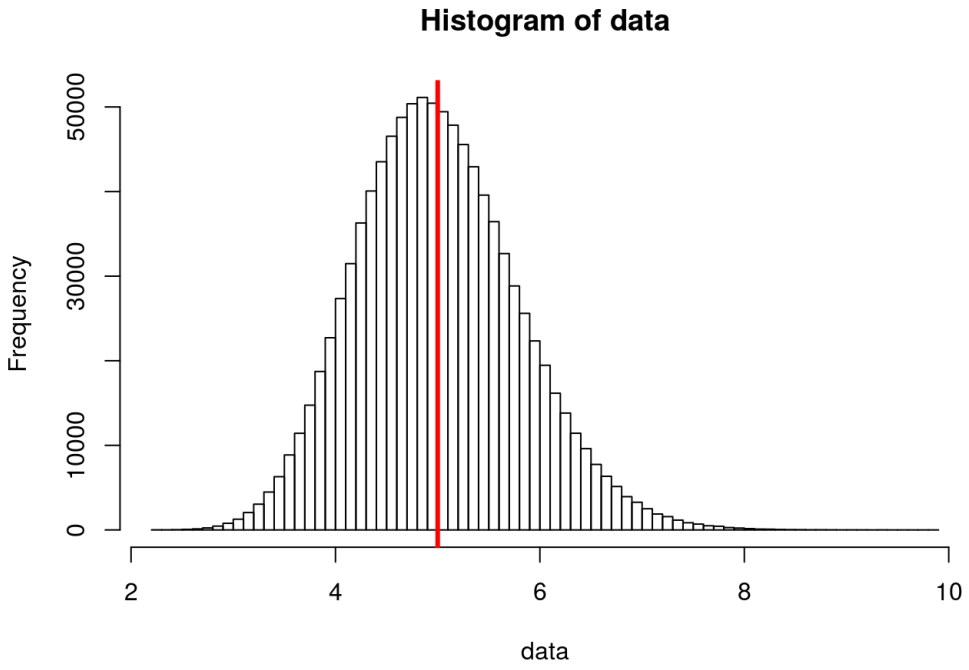
```
## [1] 4.998668 5.001768
```

True mean of 5 belongs to this interval.

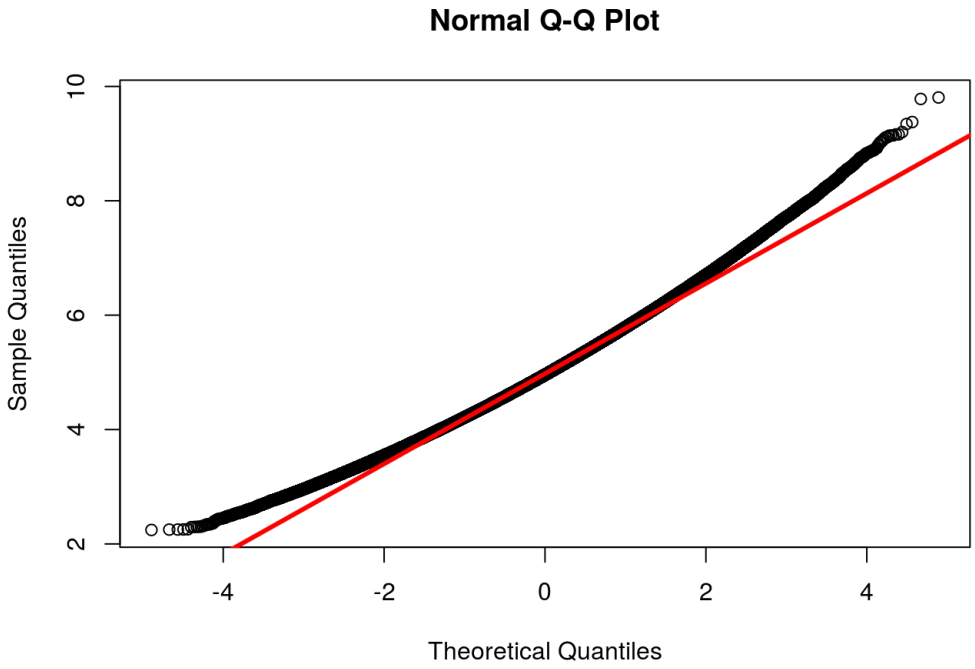
Compare with normal distribution

Let's look at the histogram and `Q-Q` plot first.

```
hist(data, breaks = 100)
abline(v = 5, lwd = 3, col = 'red')
```




```
qqnorm(data)
qqline(data, lwd = 3, col = 'red')
```



`Q-Q` plot also shows that relation is 'bend down' instead of falling on a straight line. By looking at a histogram I got a fealing that it has longer right tail. Indeed, we can confirm this by counting points below and above the mean:

```
print(sprintf("Points below the mean: %f, as a percentage: %f",
              sum(data < 5), sum(data < 5) / length(data)))
```

```
## [1] "Points below the mean: 521254.000000, as a percentage: 0.521254"
```

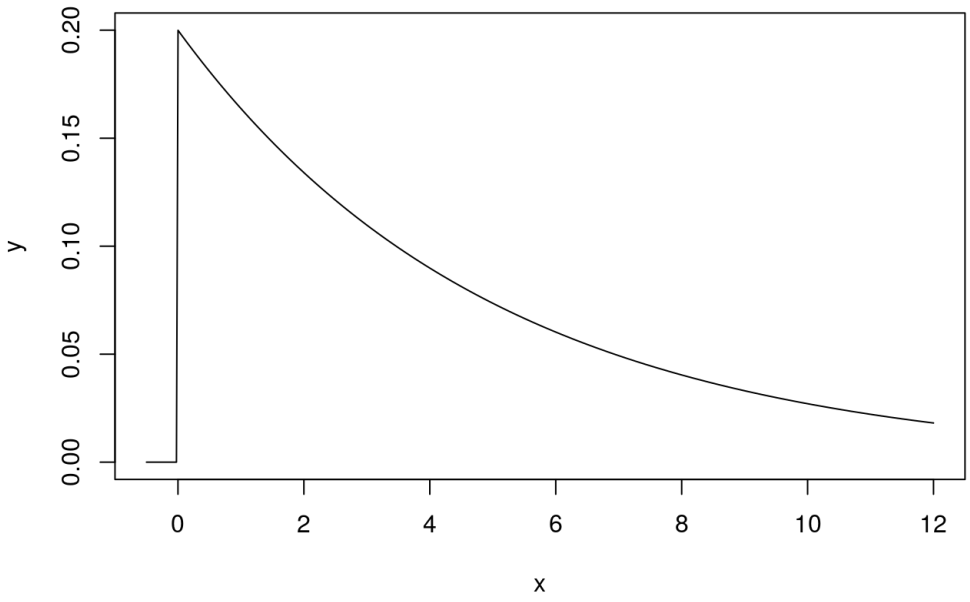
```
print(sprintf("Points above the mean: %f, as a percentage: %f",
              sum(data > 5), sum(data > 5) / length(data)))
```

```
## [1] "Points above the mean: 478746.000000, as a percentage: 0.478746"
```

Explaination of results

Exponential distribution is not symmetric about mean:

```
x <- seq(-0.5, 12, length.out=500)
y <- dexp(x, lambda)
plot(x, y, type='l')
```



It never takes values below 0, but could take an arbitrary large values. As a consequence, original data (`draw.data`) has more values below mean than above the mean:

```
v <- as.vector(as.matrix(draw.data))
print(sprintf("Points below the mean: %f, as a percentage: %f",
              sum(v < 5), sum(v < 5) / length(v)))
```

```
## [1] "Points below the mean: 25284516.000000, as a percentage: 0.632113"
```

```
print(sprintf("Points above the mean: %f, as a percentage: %f",
              sum(v > 5), sum(v > 5) / length(v)))
```

```
## [1] "Points above the mean: 14715484.000000, as a percentage: 0.367887"
```

So by taking averages of 40 values we reduced skeweness a lot, but it is still significantly skewed.

As a reminder, central limit theorem states the asympotical behavior when `n` is large enough. For this distribution `n=40` was not enough to hide the skewness.

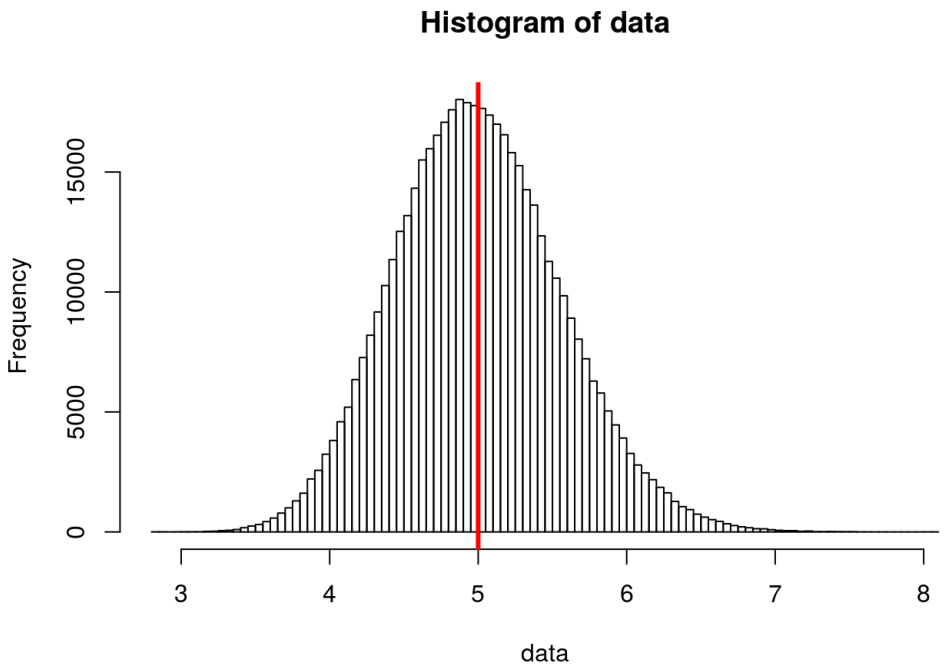
Similar distribution with bigger sample size

Let's use the same data but with `n=80` .

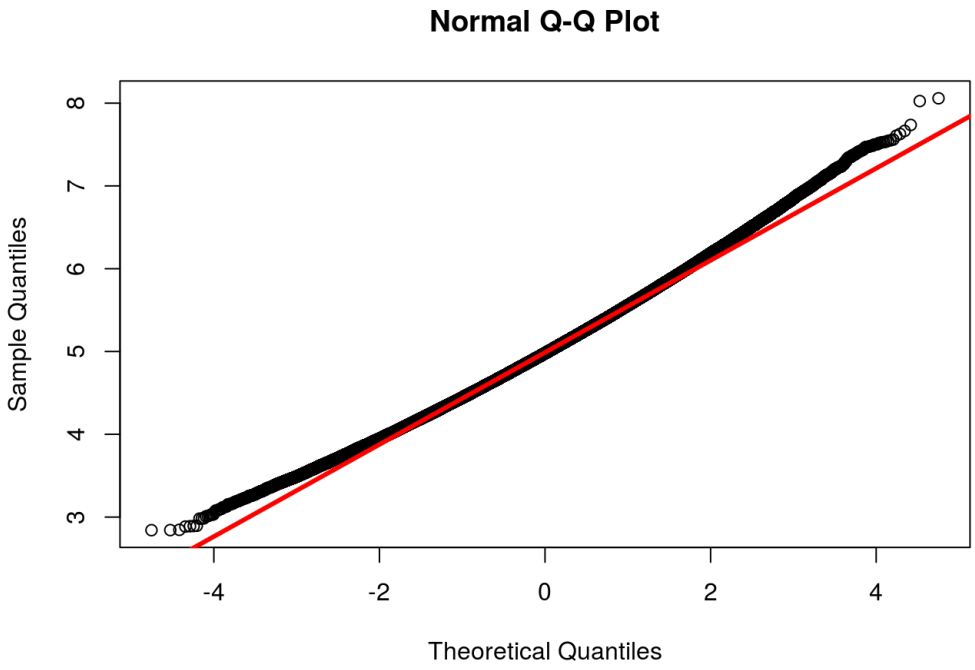
```
set.seed(12345)
```

```
number.of.simulations <- 1000000 /2
number.of.variables <- 40 * 2
lambda <- 0.2
draw.data <- as.data.frame(matrix(
  rexp(number.of.variables * number.of.simulations, lambda),
  nrow = number.of.simulations,
  ncol = number.of.variables))

data <- rowMeans(draw.data)
names(data) <- 'sample.means'
hist(data, breaks = 100)
abline(v = 5, lwd = 3, col = 'red')
```




```
qqnorm(data)
qqline(data, lwd = 3, col = 'red')
```



Almost there. `Q-Q` plot shows much better fit although we used half as many simulations.

```
print(sprintf("Points below the mean: %f, as a percentage: %f",
              sum(data < 5), sum(data < 5) / length(data)))
```

```
## [1] "Points below the mean: 257371.000000, as a percentage: 0.514742"
```

```
print(sprintf("Points above the mean: %f, as a percentage: %f",
              sum(data > 5), sum(data > 5) / length(data)))
```

```
## [1] "Points above the mean: 242629.000000, as a percentage: 0.485258"
```

We also see some improvement, although I expected better results.