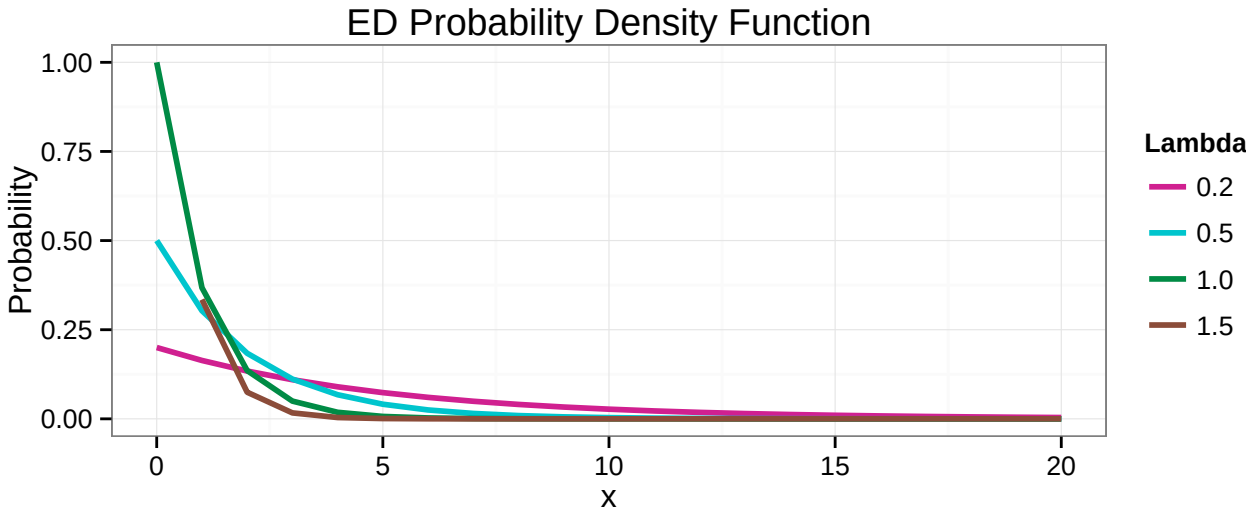
Statistical Inference: The Study of the Exponential Distribution, A Simulation Exercise

Alexander Tuzhikov

September 14, 2015

1 Overview: Exponential Distribution

In accordance with Wikipedia, **e**xponential **d**istribution (ED) *is the probability distribution that describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate.* Both *mean* and *standard deviation* of the ED is *1/lambda.* As suggested in the study objective, here we will use **lambda = 0.2**. However, for the purpose of introduction, let's reconstruct the wiki plots of the ED with different lambda:



It is obvious that ED is a skewed distribution, which drastically differes from the standard normal bell-shaped curve. We can always double check if the meand and standard deviation are indded equal in our example (see **Code Block 1**, for the plot see **Code Block 2**).

2 Simulations: Sampling the ED

Now, let's move to the first task:
1. Show the sample mean and compare it to the theoretical mean of the distribution. First we gonna need 1000 samples of size 40 form ED. We will generate a matrix of 40 rows by 1000 columns, calculate the means and store them in a vector for further reuse (see **Code Block 3**)

3 Sample Mean versus Theoretical Mean

Ok, now let's compare the theoretical mean, which is **1/lambda**, to that of the simulation procedure:

```
ed.mean.theo<- 1/lambdas[1]
ed.mean.sim<- mean(samples.colMeans) #calculate the total mean
print(c(theoretical.mean= ed.mean.theo, simulation.mean=ed.mean.sim))
```

```
## theoretical.mean  simulation.mean
##         5.000000         5.016687
```

The difference is neglegible in our case.

4 Sample Variance versus Theoretical Variance

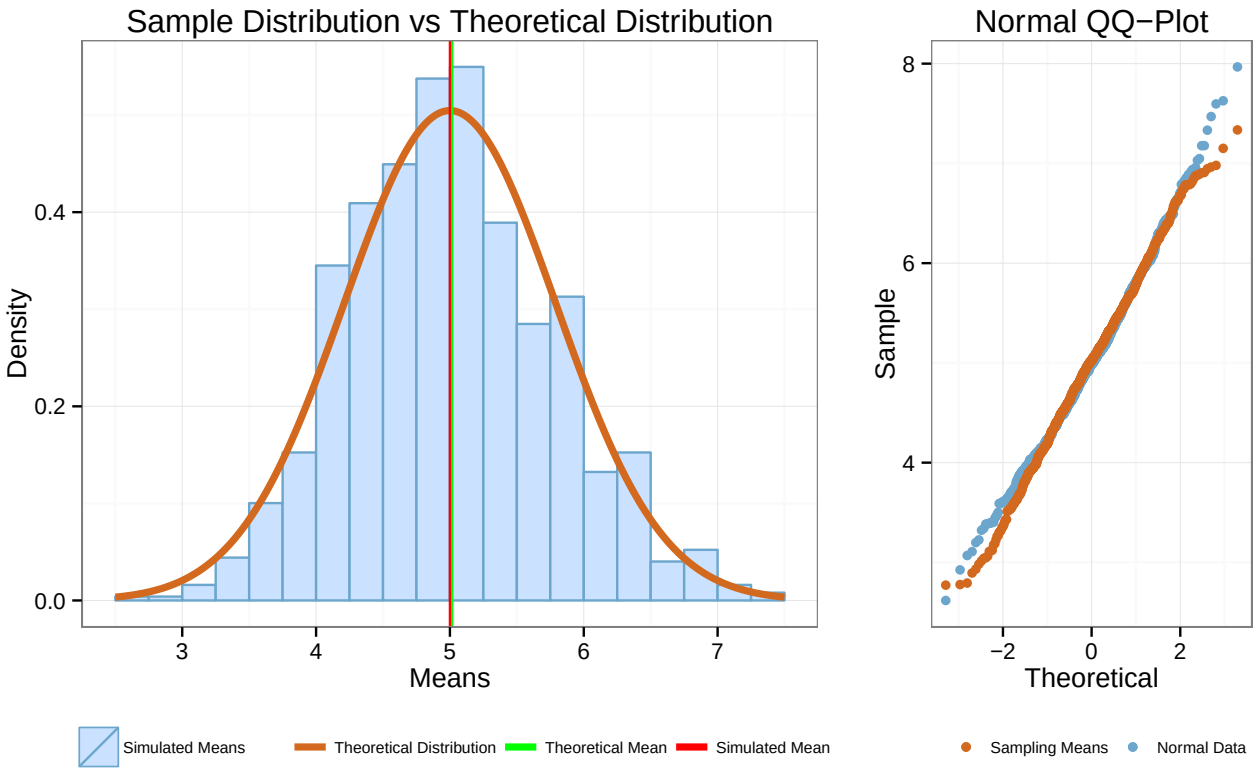
Now we will do the same in order to calculate the variance and see if it differs from the theoretical variance, as being asked in the second task: *2. Show how variable it is and compare it to the theoretical variance of the distribution.*

```
ed.var.sim<- var(colMeans(samples))#simulated variance
ed.var.theo<- (1/lambdas[1]/sqrt(n))^2#the theoretical variance is
print(c(thoretical.var=ed.var.theo, simulated.var=ed.var.sim))
```

```
## thoretical.var  simulated.var
##       0.625000       0.605856
```

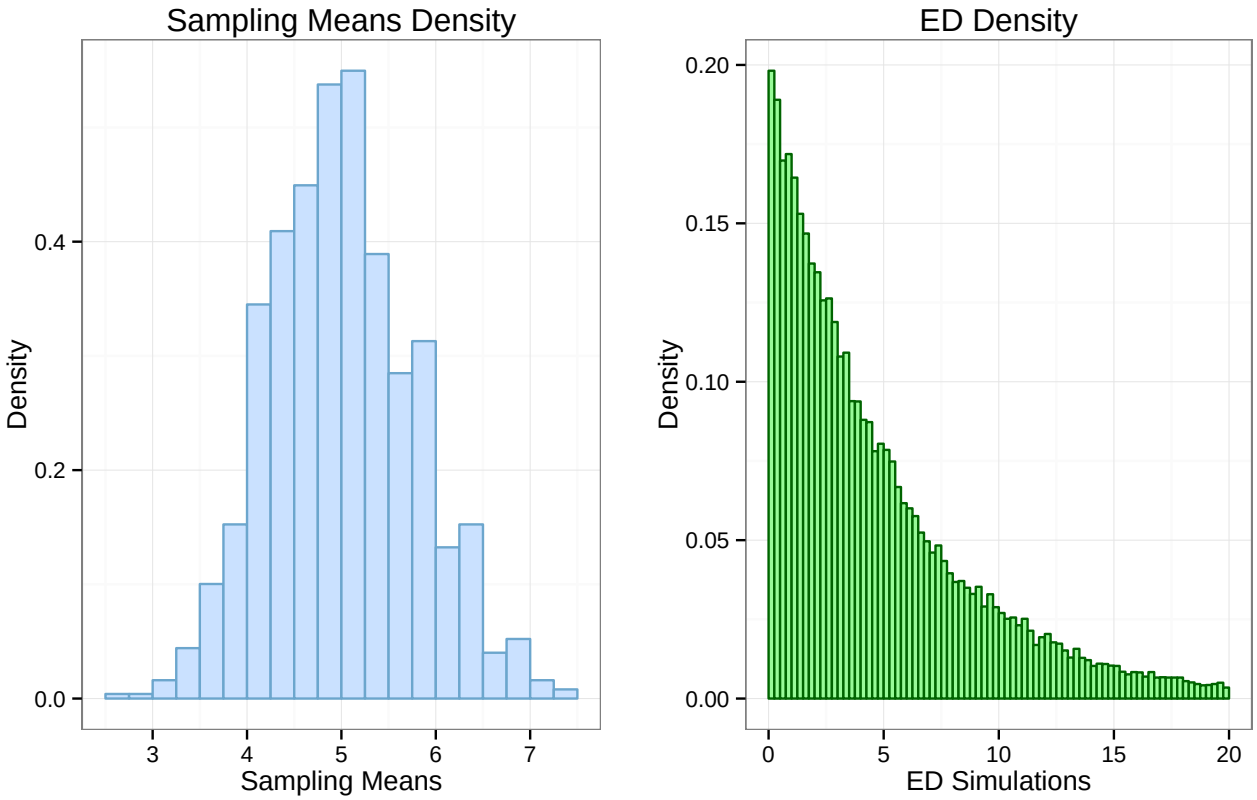
5 Distribution: Sampling Means Are Distributed Approximately Normaly

Now we move to the third task: *3. Show that the distribution is approximately normal.*



The above two plots demonstrate that the simulated meand are very close to be distributed normally. Finally, let's see how *the*

distribution of a large collection of random exponentials and the distribution of a large collection of averages of 40 exponentials differ (see **Code Block 4**)



The above plot shows how the sampling means becomes more and more "Normal distribution"-like shaped with the increase in the number of samples taken, while the density of ED becomes strongly non-normal (see **Code Block 5**).

6 Related R Code

6.1 Code Block 0

```
### Code block 1: libraries
library(dplyr)
library(ggplot2)
library(reshape2)
```

6.2 Code Block 1

```
ed.mean<- mean(rexp(1e6, 0.2)) #generate values from ED
ed.sd<- sd(rexp(1e6, 0.2))
all.equal(ed.mean, ed.sd, tolerance = 1e-2) #equal up to 1e-2 level of prescision
```

6.3 Code Block 2

```
### Code block 2: ED with different lambdas
#prepare a data.frame for the plot, melt by x, plot as line
ed.plot.df<- as.data.frame(cbind(
```

```
        x=0:40,
        la.0.2=dexp(x=0:40, lambdas[1]),
        la.0.5=dexp(x=0:40, lambdas[2]),
        la.1=dexp(x=0:40, lambdas[3]),
        la.1.5=dexp(x=0:40, lambdas[4])
)) %>%
        melt(id.vars="x") %>%
        ggplot(data=., mapping=aes(x=x, group=variable, y=value, color=variable)) +
        geom_line(size=1) + theme_bw() + xlim(0,20) + ylim(0,1) +
        labs(title="ED Probability Density Function") + ylab("Probability") +
        scale_color_manual(values=c("violetred", "turquoise3", "springgreen4",
                                    "salmon4"), labels=c("0.2","0.5", "1.0", "1.5"),
                           name="Lambda")+
        theme(legend.key = element_rect(colour = NA))
plot(ed.plot.df)
```

6.4 Code Block 3

```
### Code block 3: sampling
#sampling
samples<- as.data.frame(do.call(what = "cbind",
                                args = lapply(1:sampling.count,
                                              function(x){return(rexp(n, lambdas[1]))})))
samples.colMeans<- colMeans(samples) #calculate the column means
```

6.5 Code Block 4

```
#generate normally distributed data and combine both menas and the generated data
as.data.frame(cbind(n=1:sampling.count, sampling.means=samples.colMeans,
                    normal.data=dnorm(seq(0.01, 10, 0.01),
                                      mean = ed.mean.theo,
                                      sd = sqrt(ed.var.theo)),
                    normal.prob=rnorm(1000,
                                      mean = ed.mean.theo,
                                      sd = sqrt(ed.var.theo)))) -> ed.plot.data
combined.theo.sim.plot<- ggplot() +
        geom_histogram(data=ed.plot.data,
                       mapping=aes(x=samples.colMeans,
                                   y=..density..,
                                   fill="lightsteelblue1"),
                       color="skyblue3",
                       stat="bin",binwidth=0.25)+
        geom_line(data=ed.plot.data, mapping=aes(x=seq(0.01, 10, 0.01),
                                                 y= normal.data,
                                                 color = "chocolate"),
                  size=1.5) +
        geom_vline(aes(xintercept=ed.mean.sim, color="green")) +
        geom_vline(aes(xintercept=ed.mean.theo, color="red")) +
        xlim(2.5, 7.5)+theme_bw() +
        labs(title="Sample Distribution vs Theoretical Distribution") +
        xlab("Means") + ylab("Density") +
        scale_fill_identity(name = "", guide = "legend",
```

```
                            labels = c("Simulated Means")) +
        scale_colour_manual(name = "",
                            values = c("chocolate"="chocolate",
                                       "red"="red", "green"="green"),
                            labels = c("Theoretical Distribution",
                                       "Theoretical Mean", "Simulated Mean")) +
        theme(legend.key = element_rect(colour = NA), legend.position="bottom",legend.box = "horizontal
              legend.text=element_text(size= 7))
qq.plot<- ggplot() + stat_qq(data=ed.plot.data,
                             mapping=aes(sample=sampling.means,color="skyblue3")) +
        stat_qq(data=ed.plot.data,
                mapping=aes(sample=normal.prob,color="chocolate")) +
        theme_bw() +
        xlab("Theoretical") + ylab("Sample") + labs(title="Normal QQ-Plot") +
        scale_color_manual(name="",
                           values=c("skyblue3"="skyblue3","chocolate"="chocolate"),
                           labels=c("Sampling Means","Normal Data"))+
        theme(legend.key = element_rect(colour = NA), legend.position="bottom",
              legend.text=element_text(size= 7))

library(gridExtra)

grid.arrange(combined.theo.sim.plot, qq.plot, ncol=2,widths=c(2, 1))
```

6.6 Code Block 5

```
ed.means.hist<- ggplot() +
        geom_histogram(data=ed.plot.data,
                       mapping=aes(x=sampling.means, y=..density..),
                       fill="lightsteelblue1",
                       color="skyblue3", stat="bin", binwidth=0.25) +
        theme_bw() + xlab("Sampling Means") + ylab("Density") +
        labs(title="Sampling Means Density") +
        xlim(2.5, 7.5)+
        theme(legend.key = element_rect(colour = NA))
ed.values.hist<- ggplot() +
        geom_histogram(data=melt(samples),
                       mapping=aes(x=value, y=..density..),
                       fill="palegreen",
                       color="darkgreen",
                       stat="bin",
                       binwidth=0.25) +
        theme_bw() + xlab("ED Simulations") + ylab("Density") +
        labs(title="ED Density")+  xlim(0, 20) +
        theme(legend.key = element_rect(colour = NA))
grid.arrange(ed.means.hist, ed.values.hist, ncol=2)
```