Real_experiment_counting_cars_Exponential_Gamma_MLikelihood_and_

March 25, 2021

1 Real experiment with exponential and gamma distribution

@Author: Javier Cebrián Casado

This experiment is made with real data collected by me on March 23, 2021 between 12:30 and 13:00 on the M-40 road in Madrid.

It consists of taking the times in which white cars passed.

The objective is to check whether this type of random variables can be modeled with exponential distributions or more generally with gamma type.

[1]:
```
import pandas as pd
import numpy as np
import re
import plotly.express as px
import plotly.graph_objects as go
from scipy import stats

import plotly.io as pio
pio.renderers.default = "notebook+pdf"
```

2 1- Read data

timer.txt is the file generated with the ouput of the app http://play.google.com/store/apps/details?id=uk.co.dedmondson.timer.split

There are two columns: first is the time stamps and the second is the interval between timestamps. In this case I use two regular expression to extract this data and create two arrays (timestamps and intervals) in seconds.

[2]:
```
filename = 'timer.txt'
pattern  = '(\d{2}:\d{2}.\d{3})\s{3}(\d{2}:\d{2}.\d{3})'
pattern2 = '(\d{2}):(\d{2}.\d{3})'

timestamps = np.empty(392)
intervals = np.empty(392)
```

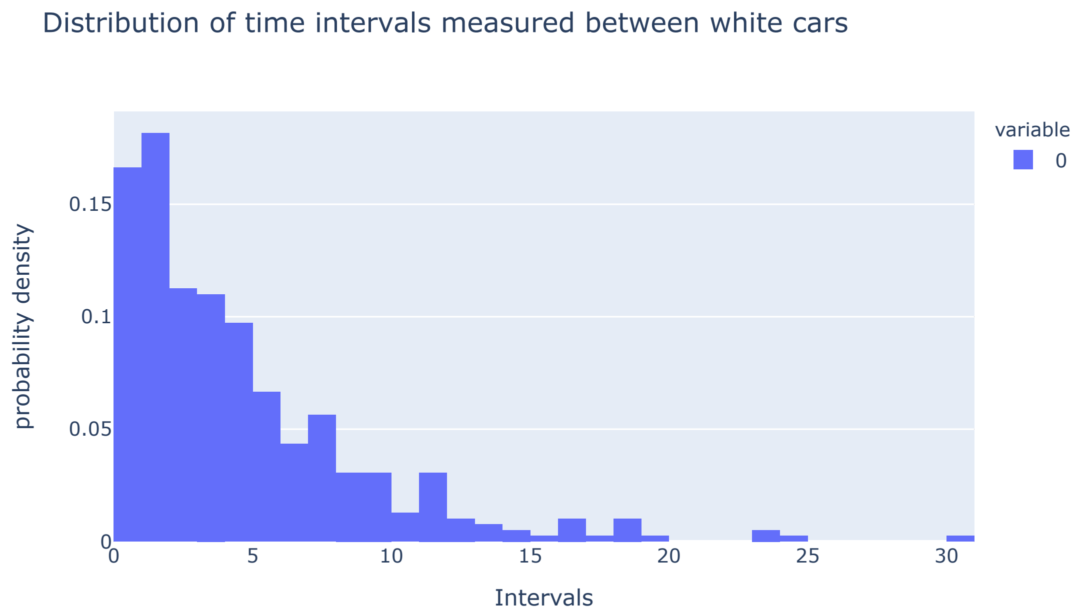
```
# Make sure file gets closed after being iterated
with open(filename, 'r') as f:
   # Read the file contents and generate a list with each line
   lines = f.readlines()

f.close
lines

for i,_ in enumerate(lines):
    if len(re.findall(pattern,lines[i]))>0:
        t=re.findall(pattern,lines[i])[0][0]
        convert=float(re.findall(pattern2,t)[0][0])*60+float(float(re.
 ↪findall(pattern2,t)[0][1]))
        timestamps[i]=convert

        t=re.findall(pattern,lines[i])[0][1]
        convert=float(re.findall(pattern2,t)[0][0])*60+float(float(re.
 ↪findall(pattern2,t)[0][1]))
        intervals[i]=convert

timestamps=np.flip(timestamps)
intervals=np.flip(intervals)
print('Number of white cars per second: '+str(len(timestamps)/(30*60)))
print('Total number of white cars in 30 minutes: '+str(len(timestamps)))
```

```
Number of white cars per second: 0.21777777777777776
Total number of white cars in 30 minutes: 392
```

2.1 2- Data exploration

Histogram of time intervals between events (white cars)

[3]:
```
fig = px.histogram(intervals[:-1],histnorm='probability density',width=800,␣
 ↪height=320,title='Distribution of time intervals measured between white␣
 ↪cars')
fig.update_xaxes(title='Intervals')
fig.show()
```



With a theory book in hand, the distribution is expected to be roughly exponential:

$$f(x, \lambda) = \lambda e^{-\lambda x}$$

Where $\lambda > 0$ and represents the rate parameter, withe cars per second in this case.

Anyway this is an special case of gamma function so also is going to be include in the analysis:

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

Where $\beta = \frac{1}{\lambda}$, $x >= 0$, $a>0 $ and $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$

2.2 3- Maximum likelihood estimation

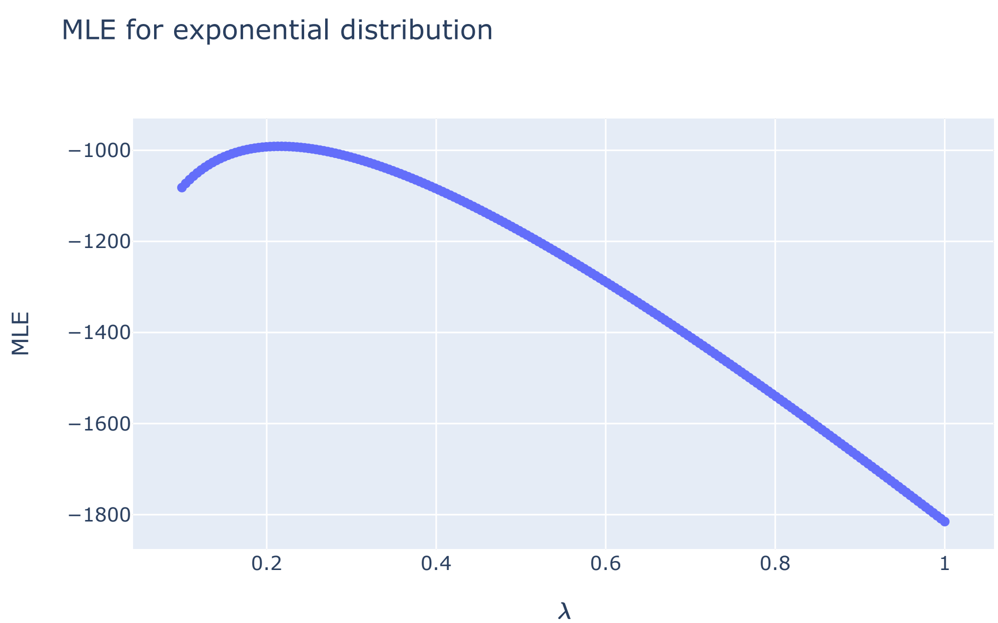
Although the value that λ represents in this case we have already found ($\lambda = 0.2178[\frac{cars}{s}]$), I am going to pose the problem on the basis that it is unknown. So applying maximum likelihood I will see if I get a similar λ result.

Estimate the parameters $\Theta - \{\theta_1, \theta_2, ..., \theta_m\}$ that describes $f(X)$ probability density function of n i.i.d. random variables $x_1, x_2 ..., x_n$ by maximizing likelihood function $f(X|\Theta) = L(\Theta|X)$:

$$\hat{\Theta} = \frac{argmax}{\theta \in \Theta} \hat{L_n}(\Theta|x_i) = \frac{argmax}{\theta \in \Theta} \prod_i^n f_i(x_i|\Theta)$$

So equivalently applying the properties of logarithms to facilitate the calculation $\underset{\theta \in \Theta}{argmax} \sum_i^n ln(f_i(x_i|\Theta))$

[4]:
```
lamb = np.linspace(0.1, 1, 200, endpoint=True)
L=np.empty([len(lamb)])
for i in range(len(lamb)):
    L[i]=(np.log(stats.expon.pdf(intervals[:-1],scale=1/lamb[i])).sum())

fig = px.scatter(x=lamb,y=L,width=800, height=320,title='MLE for exponential␣
 ↪distribution')
fig.update_xaxes(title='$\lambda$')
fig.update_yaxes(title='MLE')
fig.show()
expMaxLambda=lamb[np.argmax((L))]
print('Estimated Lambda = '+str(expMaxLambda))
```



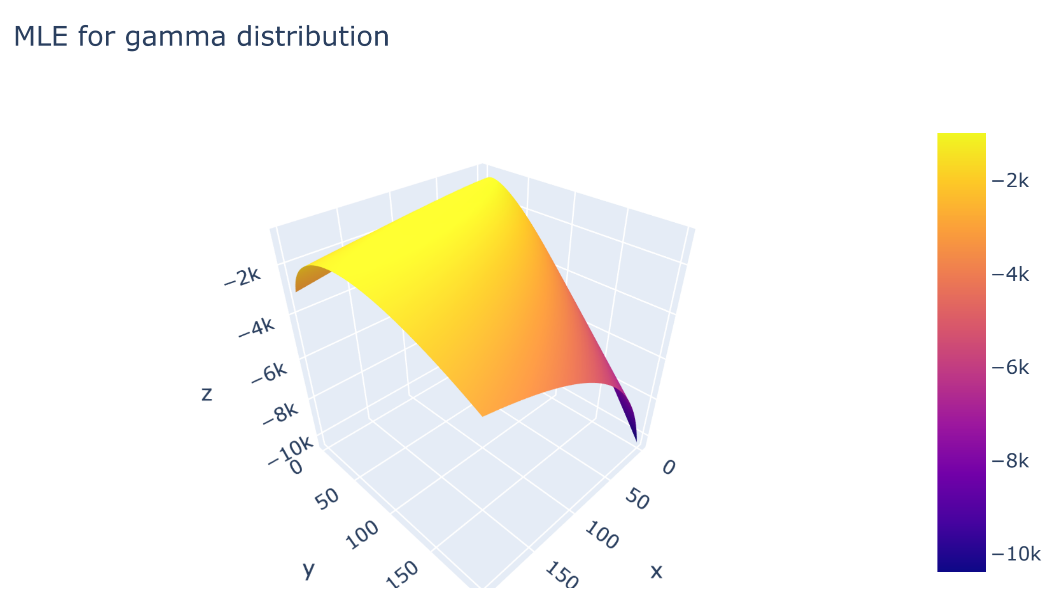

```
Estimated Lambda = 0.21758793969849247
```

As can be seen estimated lambda is in practice the measured value $\lambda = 0.2178[\frac{cars}{s}]$

Now, MLE for gamma distribution:

[5]:
```
av = np.linspace(0.1, 10, 200, endpoint=True)
lamb = np.linspace(0.1, 1, 200, endpoint=True)
```

```
l=np.empty([len(av), len(lamb)])
for i in range(len(av)):
    for j in range(len(lamb)):
        l[i,j]=(np.log(stats.gamma.pdf(intervals[:-1],a=av[i],scale=1/lamb[j])).
 ↪sum())

fig = go.Figure(data=[go.Surface(z=l)])
fig.update_layout(title='MLE for gamma distribution', autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))

fig.update_xaxes(title='a')
fig.update_yaxes(title='$\lambda$')

fig.show()

gammaMaxA=av[np.argmax(np.max(l, axis=1))]
gammaMaxLambda=lamb[np.argmax(np.max(l, axis=0))]

print('Estimated a: '+str(round(gammaMaxA,4)))
print('Estimated lambda: '+str(round(gammaMaxLambda,4)))
```

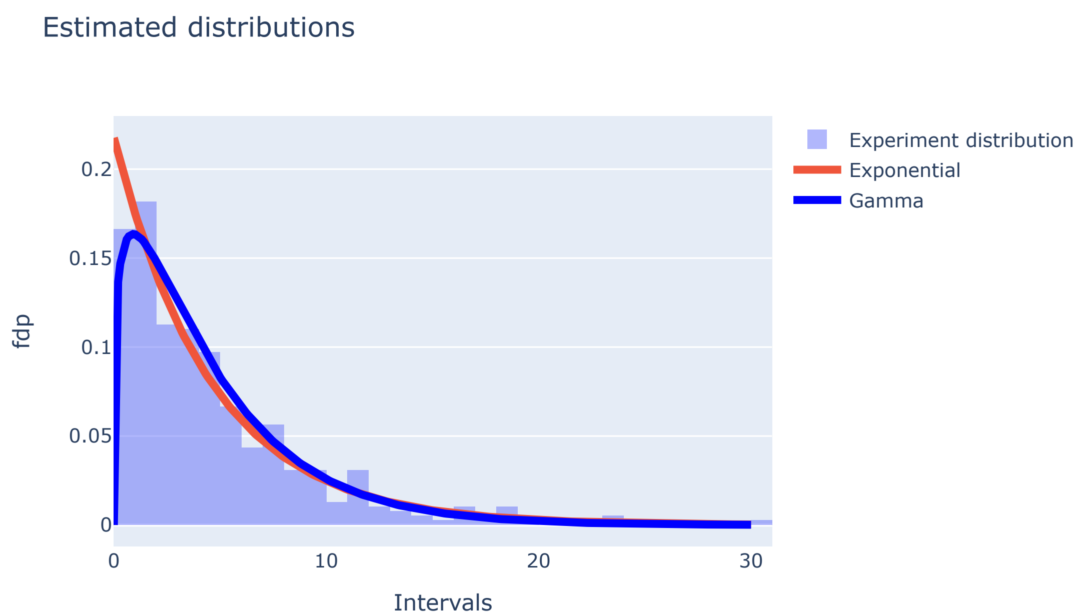



```
Estimated a: 1.2442
```

Estimated lambda: 0.2673

2.3 Plot of distributions

[6]:
```
x=np.linspace(0,30,300 )

data0 = go.Histogram(x=intervals[:-1], histnorm='probability␣
↪density',name='Experiment distribution',opacity=0.5)
data1 = go.Scatter(x=x, y=stats.expon.pdf(x,scale=1/expMaxLambda), mode =␣
↪'lines', name='Exponential',line=dict(width=5))
data2 = go.Scatter(x=x, y=stats.gamma.pdf(x,a=gammaMaxA,scale=1/
↪gammaMaxLambda), mode = 'lines',␣
↪name='Gamma',line=dict(color='blue',width=5))
data = [data0, data1,data2]
layout = go.Layout(title='Estimated distributions')
fig = go.Figure(data= data, layout = layout)
fig.update_xaxes( title='Intervals')
fig.update_yaxes(title='fdp')
fig.show()
```



2.4 4- Nonparametrical Anderson-Darling test

Measures the significance level. Null-hypothesis = Distributions are equal

4.1 Exponential

[7]:
```python
# stats.anderson_ksamp([intervals[:-1],
# np.random.exponential(scale=1/expMaxLambda, size=len(intervals[:-1]))])

stats.ks_2samp(intervals[:-1], np.random.exponential(scale=1/expMaxLambda,␣
 ↪size=len(intervals[:-1])))
```

[7]: KstestResult(statistic=0.058823529411764705, pvalue=0.5085813922360893)

4.2 Gamma

[8]:
```python
# stats.anderson_ksamp([intervals[:-1],
# np.random.gamma(shape=gammaMaxA,scale=1/gammaMaxLambda, size=len(intervals[:
 ↪-1]))])

stats.ks_2samp(intervals[:-1], np.random.gamma(shape=gammaMaxA,scale=1/
 ↪gammaMaxLambda, size=len(intervals[:-1])))
```

[8]: KstestResult(statistic=0.0741687979539642, pvalue=0.23258198119900275)

In neither case can the null hypothesis be rejected