Equação Logística $\rho = 4.00$, $A_0 = 0.0001$

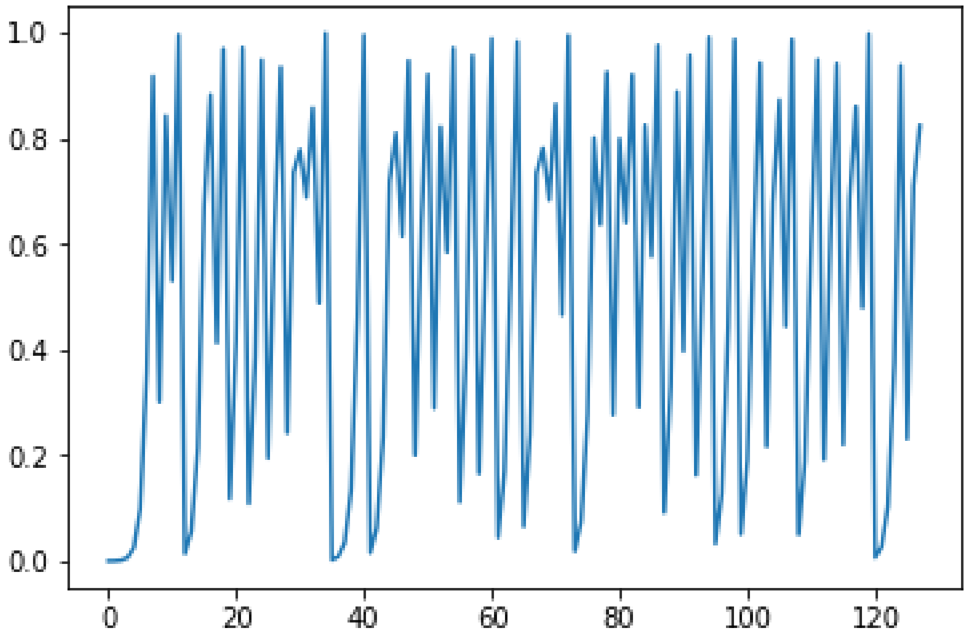
27 de maio de 2018

```
In [1]: from scipy.stats import moment
        from scipy.stats import kurtosis, skew, scoreatpercentile
        from scipy.stats import norm, lognorm, beta
        from scipy.optimize import minimize

        from numpy import zeros, fromiter, savetxt
        from IPython.display import Image

        import subprocess

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        import auxiliar_matcomp as aux

        #%matplotlib inline

        size = 2**12
        t = fromiter((i for i in range(0,size)), int, size)
```

1 Série Completa

1.1 Gerando série temporal e plotando resultado

```
In [2]: name = "A.ex:1.1.b"

        rho = 4.00

        A = zeros(size)
        A[0] = 0.0001

        for i in range(0, size-1):
            A[i+1] = rho*A[i]*(1-A[i])

        savetxt(name + ".txt", A)

        save_A = A

In [3]: num_points = 128
        plt.plot(t[0:num_points], A[0:num_points])
        plt.show()
```



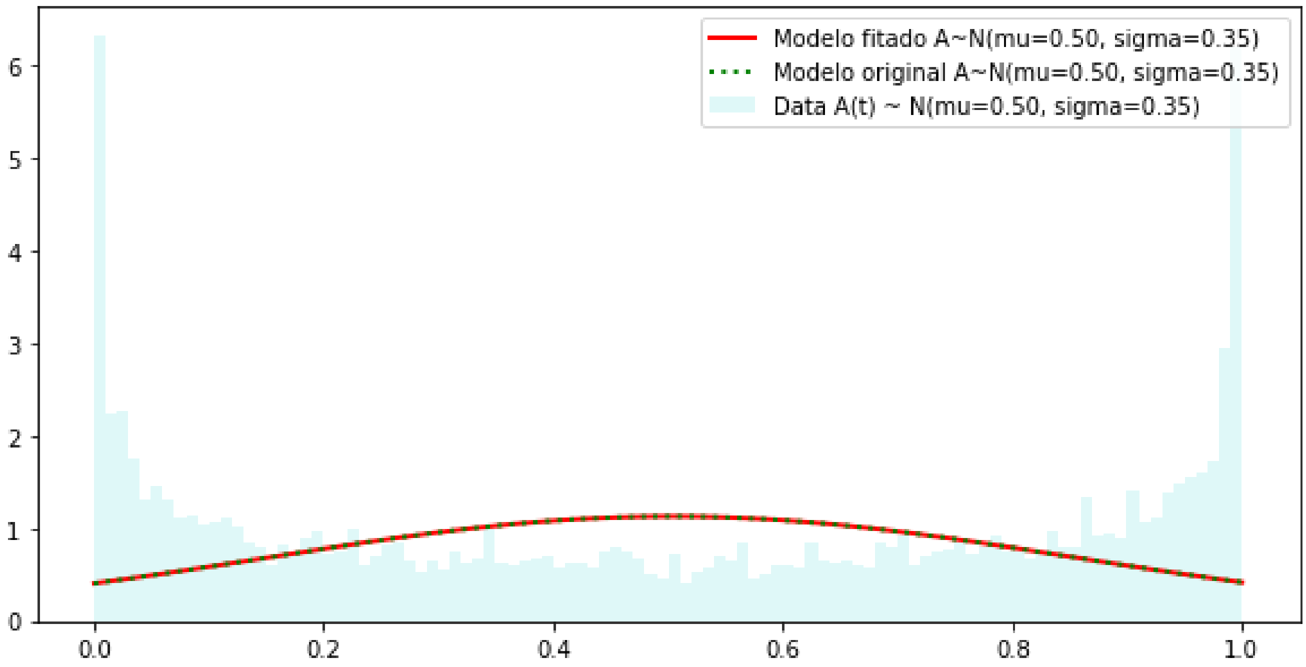
1.2 Calculando os momentos do ensemble

```
In [4]: A_mean, A_var, A_skew, A_kurtosis = aux.calcMoments(A)

        print("mean : ", A_mean)
        print("var  : ", A_var)
        print("skew : ", A_skew)
        print("kurt : ", A_kurtosis)

        A_Q1 = scoreatpercentile(A, 25)
        A_Q3 = scoreatpercentile(A, 75)

        print("Q1   : ", A_Q1)
        print("Q3   : ", A_Q3)
```

```
mean :  0.50337804739
var  :  0.124136207879
skew :  -0.0176098581162
kurt :  -1.49632517826
Q1   :  0.151266836474
Q3   :  0.854218928748
```

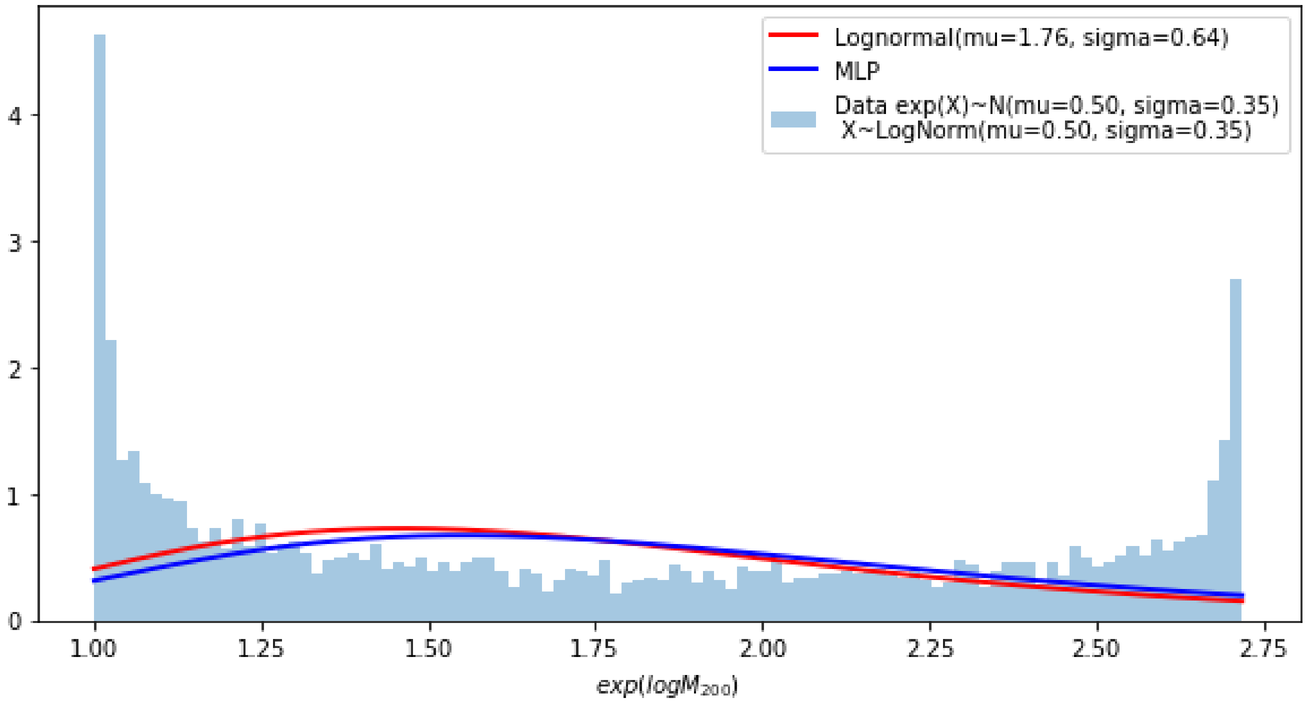
1.3 Fitando uma distribuição normal

```
In [5]: aux.fitting_normal_distribution(A)
```



1.4 Fitando uma distribuição lognormal

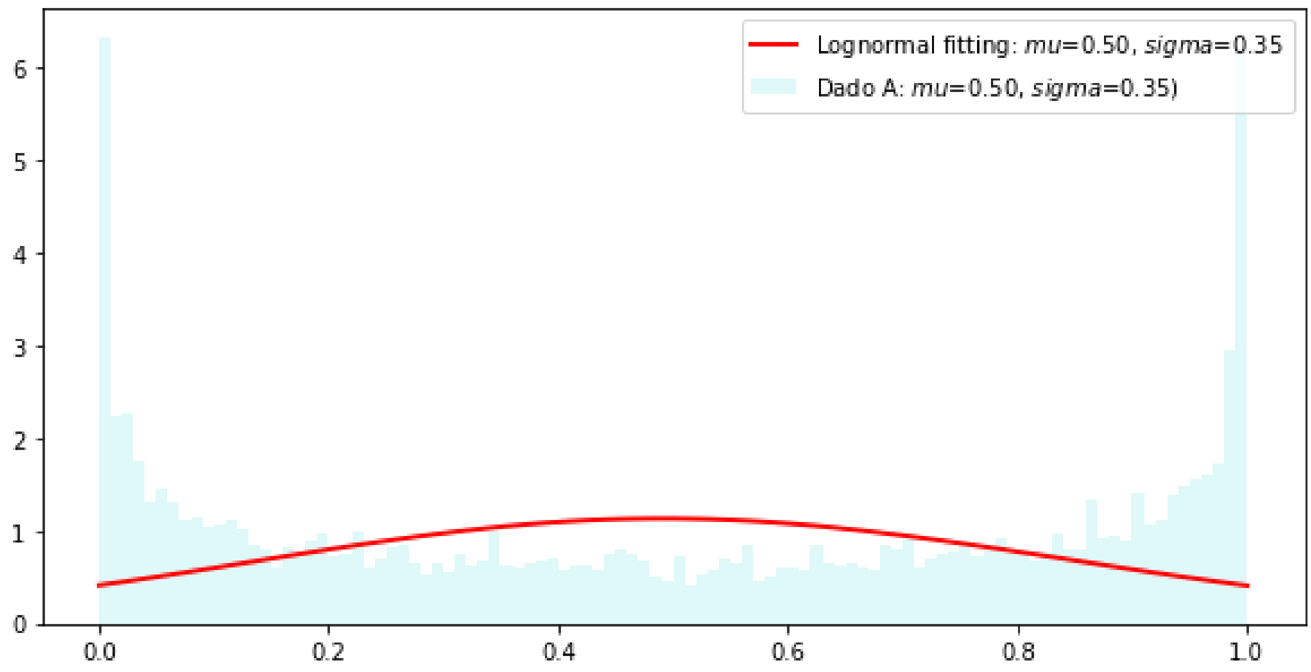
```
In [6]: aux.fitting_lognormal_and_mlp_distribution(A)
```



1.5 Fitando uma distribuição lognormal (utlizando minha implementação)

```
In [7]: aux.fitting_lognormal_distribution(A)
```

```
parametros de fitting:  (0.02245750779438966, -15.170544981196269, 15.669430642098931)
        Fitado                         Original
mean :  0.5028375167836323          0.50337804739
var   :  0.1239247648490273          0.124136207879
skew :  0.06739234943154261          -0.017609858116157458
kurt :  0.008075287783896101          -1.496325178258426
```

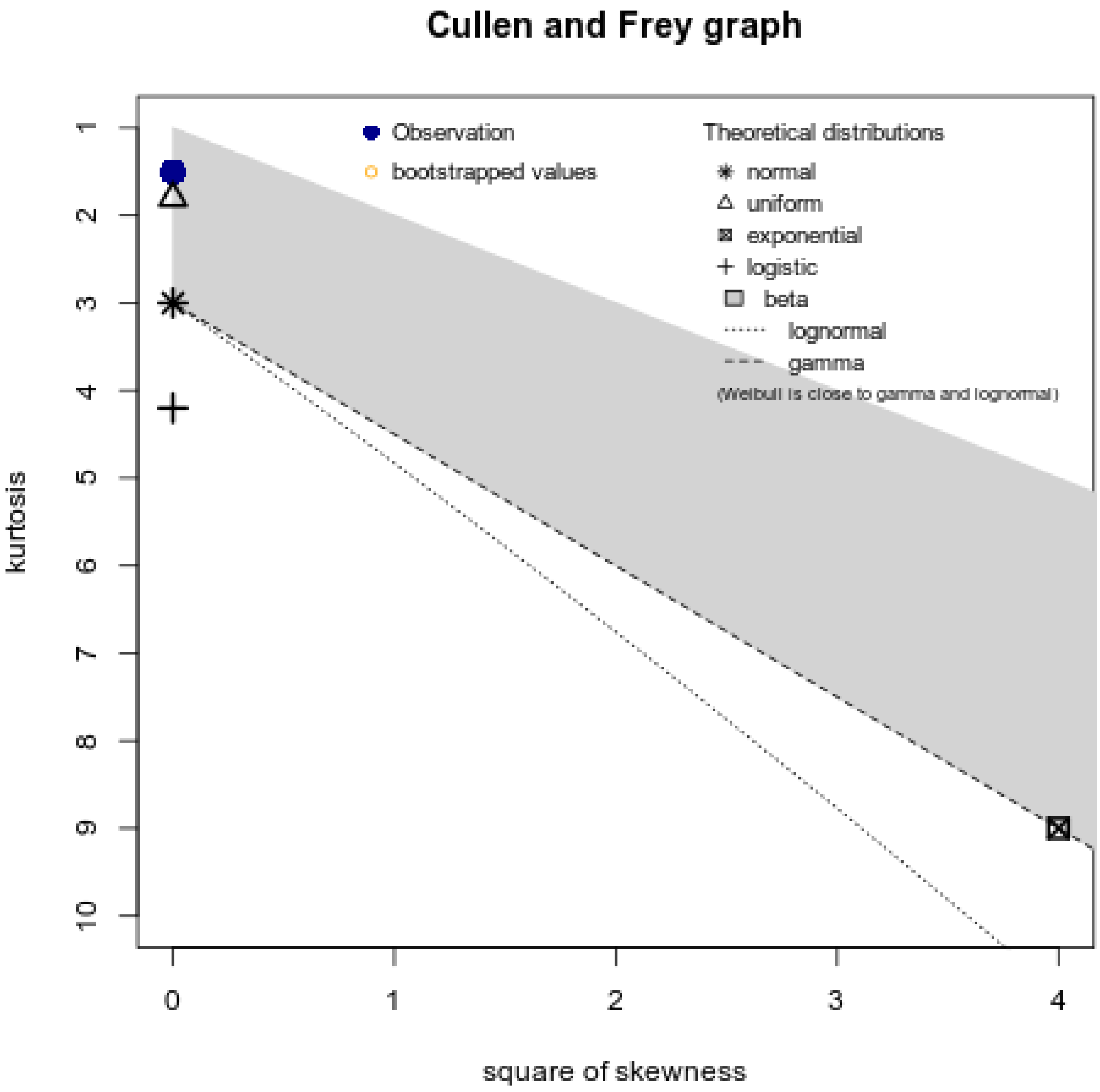


1.6 Plotando dados no espaço de Cullen-Frey

```
In [8]: command = 'Rscript'
        path_script = 'cullen_frey_script.R'

        # define arguments
        args = [name,]

        # build subprocess command
        cmd = [command, path_script] + args

        x = subprocess.check_output(cmd, universal_newlines=True)
        print(x)

        Image(name+".png")
```

```
summary statistics
------
min:  2.495392e-07   max:  0.9999999
median:  0.5031523
mean:  0.503378
estimated sd:  0.3523727
estimated skewness:  -0.01761631
estimated kurtosis:  1.503313
```

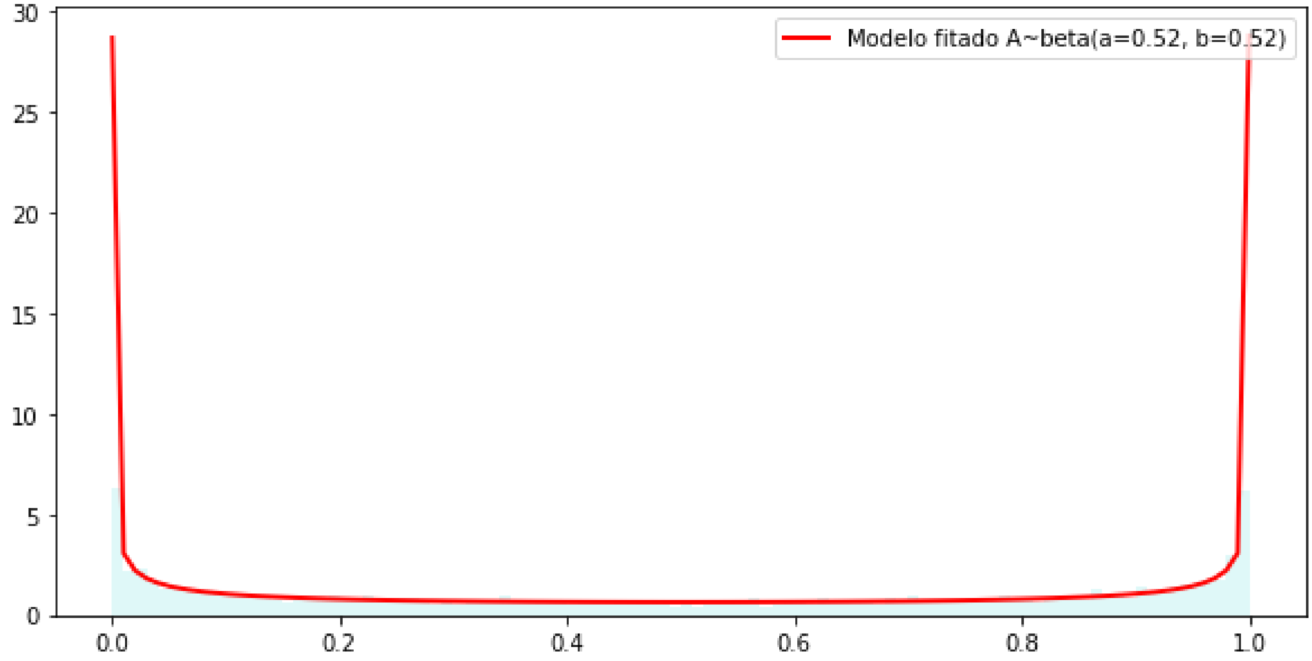
Out[8]:



1.7 Fitando melhor distribuição segundo método de Cullen-Frey

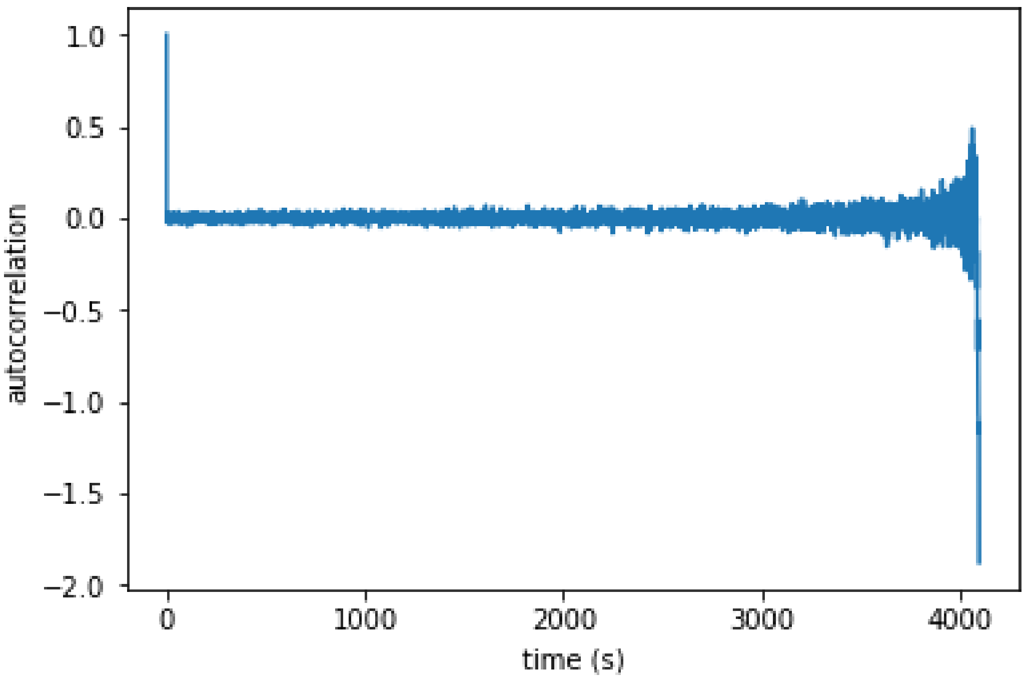
```
In [9]: aux.fitting_beta_distribution(A)
```

```
parametros de fitting:  (0.5161195993524750 9, 0.5156755552728162 8, -9.9750460756089341e-05, 1.0001996880
        Fitado                         Original
mean :  0.5002153168935751          0.50337804739
var  :  0.12309302299318188          0.124136207879
skew :  -0.0008093432279352365          -0.017609858116157458
kurt :  -1.4881701056196341          -1.496325178258426
```



1.8 Calculando autocorrelação

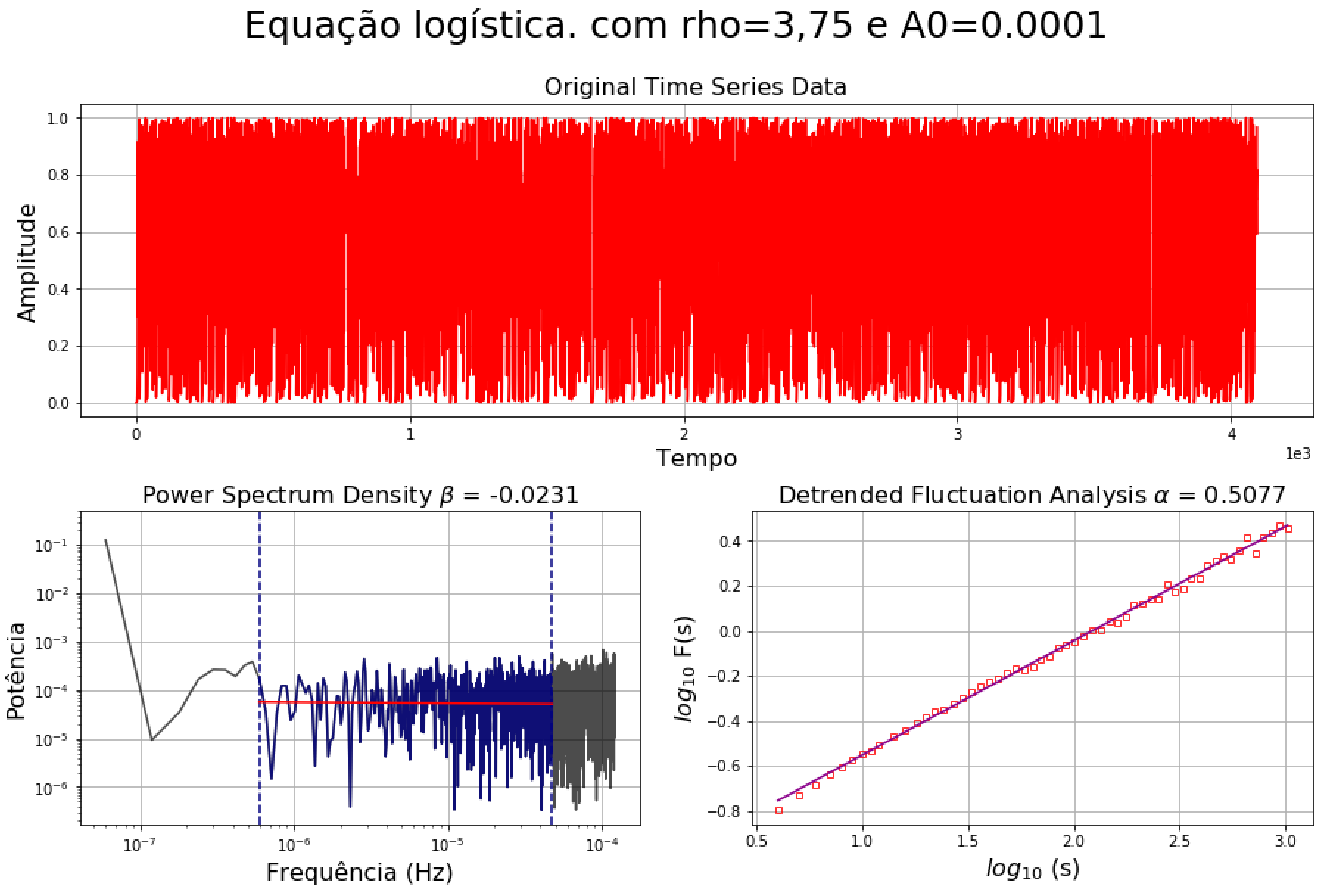
```
In [10]: aux.plot_estimated_autocorrelation(t, A, 0, len(A))
```



1.9 Plotando DFA e PSD

```
In [11]: aux.plot_psd_dfa(A, 'Equação logística. com rho=3,75 e A0=0.0001')
```

```
Original time series data (4096 points):

First 10 points: [  1.00000000e-04   3.99960000e-04   1.59920013e-03   6.38657075e-03
   2.53831298e-02   9.89553063e-02   3.56652615e-01   9.17806108e-01
   3.01752223e-01   8.42791276e-01]


1. Plotting time series data...
2. Plotting Power Spectrum Density...
3. Plotting Detrended Fluctuation Analysis...
```



2 Analise dos primeiros 1024 pontos

```
In [12]: A = save_A[1024:]
         name = "A.ex:1.1.b"
         savetxt(name + ".txt", A)
```

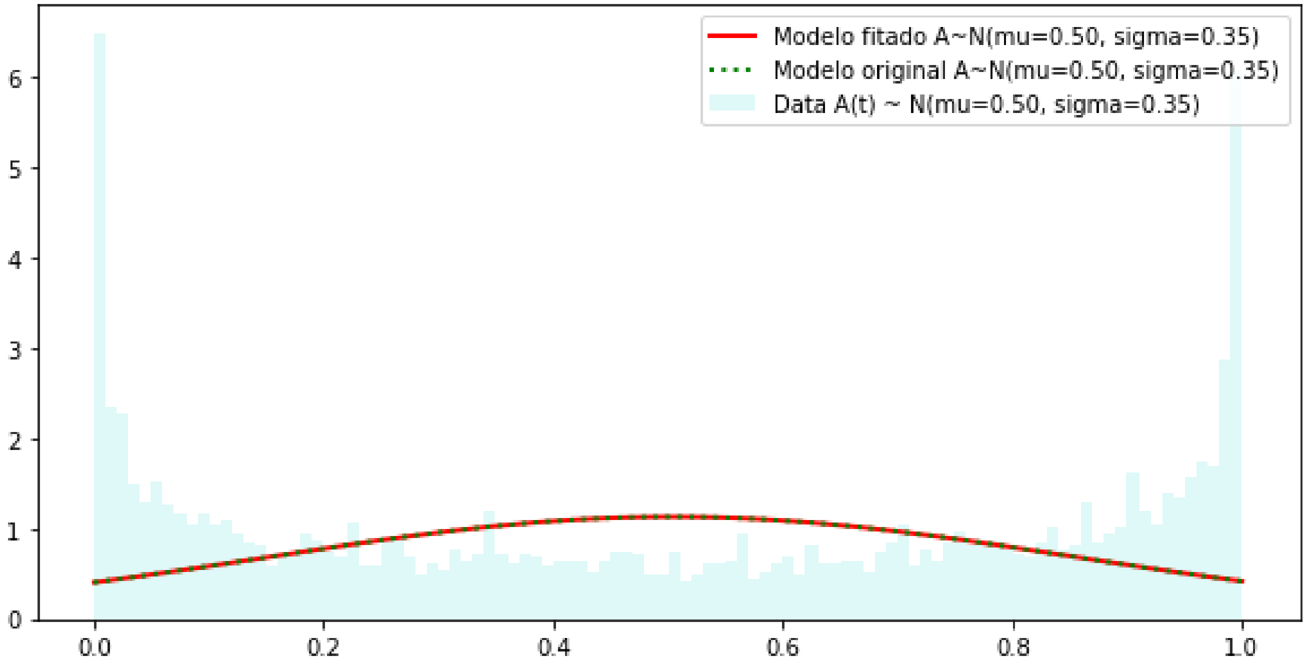
2.1 Calculando os momentos do ensemble

```
In [13]: A_mean, A_var, A_skew, A_kurtosis = aux.calcMoments(A)

         print("mean : ", A_mean)
         print("var  : ", A_var)
         print("skew : ", A_skew)
         print("kurt : ", A_kurtosis)

         A_Q1 = scoreatpercentile(A, 25)
         A_Q3 = scoreatpercentile(A, 75)

         print("Q1   : ", A_Q1)
         print("Q3   : ", A_Q3)
```

```
mean :  0.50393379815
var  :  0.1239955322
skew :  -0.0230936095474
kurt :  -1.49549055889
Q1   :  0.153616020599
Q3   :  0.854086850455
```

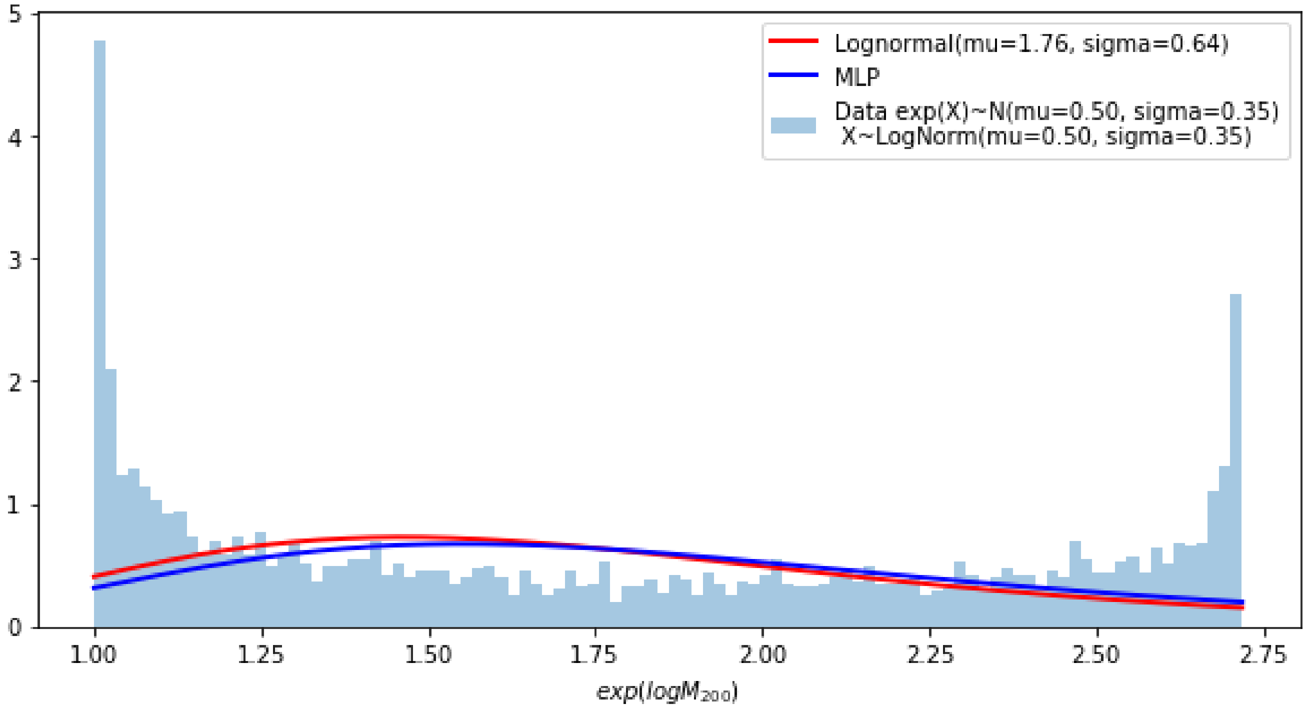
2.2 Fitando uma distribuição normal

```
In [14]: aux.fitting_normal_distribution(A)
```



2.3 Fitando uma distribuição lognormal

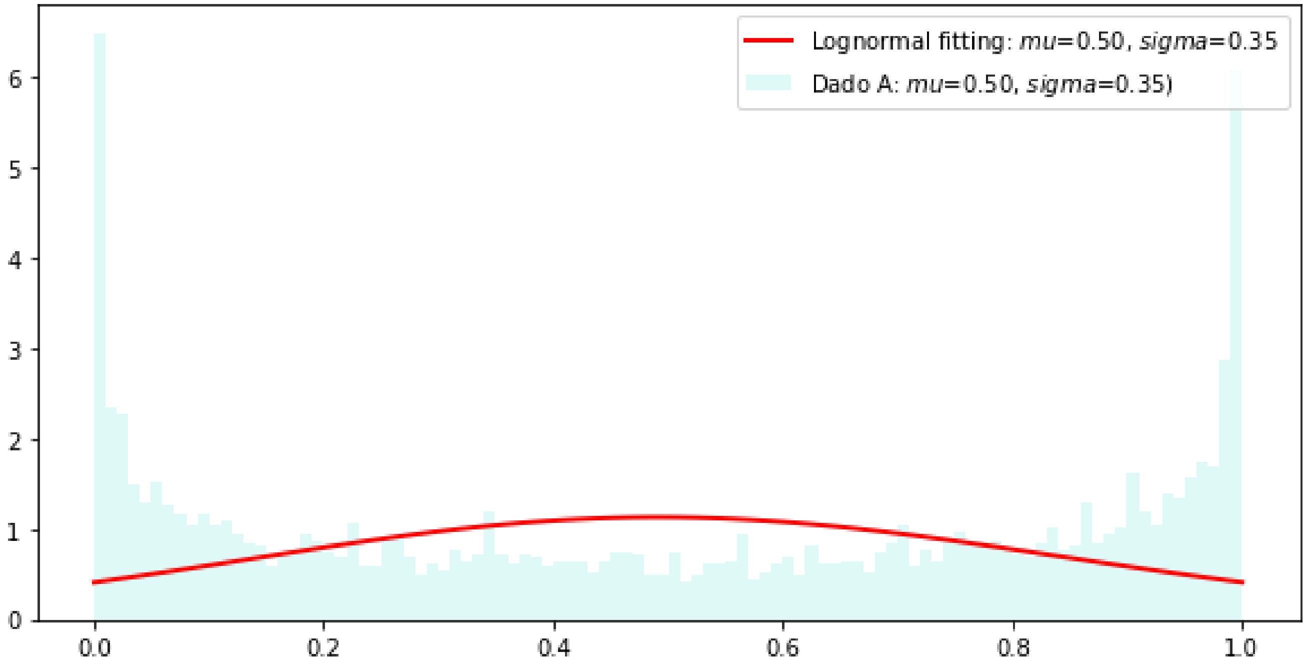
```
In [15]: aux.fitting_lognormal_and_mlp_distribution(A)
```



2.4 Fitando uma distribuição lognormal (utlizando minha implementação)

```
In [16]: aux.fitting_lognormal_distribution(A)
```

```
parametros de fitting:  (0.024836027422982222, -13.674989102840154, 14.175417798586405)
        Fitado                         Original
mean :  0.5048012691279595           0.50393379815
var  :  0.12406172966349534           0.1239955322
skew :  0.07453490007092164           -0.02309360954742383
kurt :  0.009878008660902715           -1.4954905588860032
```

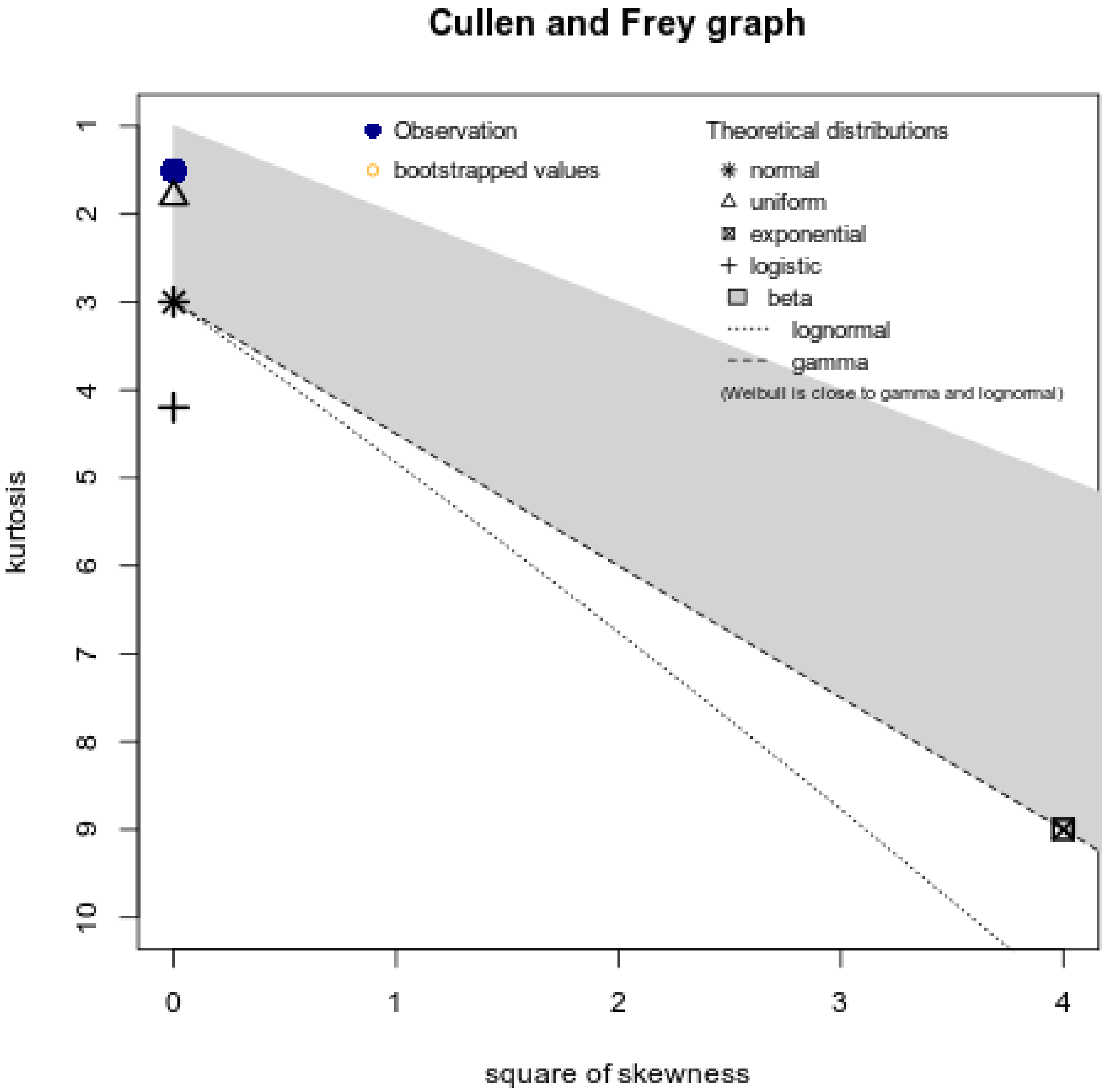


2.5 Plotando dados no espaço de Cullen-Frey

```
In [17]: command = 'Rscript'
         path_script = 'cullen_frey_script.R'

         # define arguments
         args = [name,]

         # build subprocess command
         cmd = [command, path_script] + args

         x = subprocess.check_output(cmd, universal_newlines=True)
         print(x)


         Image(name+".png")
```

```
summary statistics
------
min:  2.495392e-07   max:  0.9999999
median:  0.5035671
mean:  0.5039338
estimated sd:  0.3521873
estimated skewness:  -0.02310489
estimated kurtosis:  1.504028
```

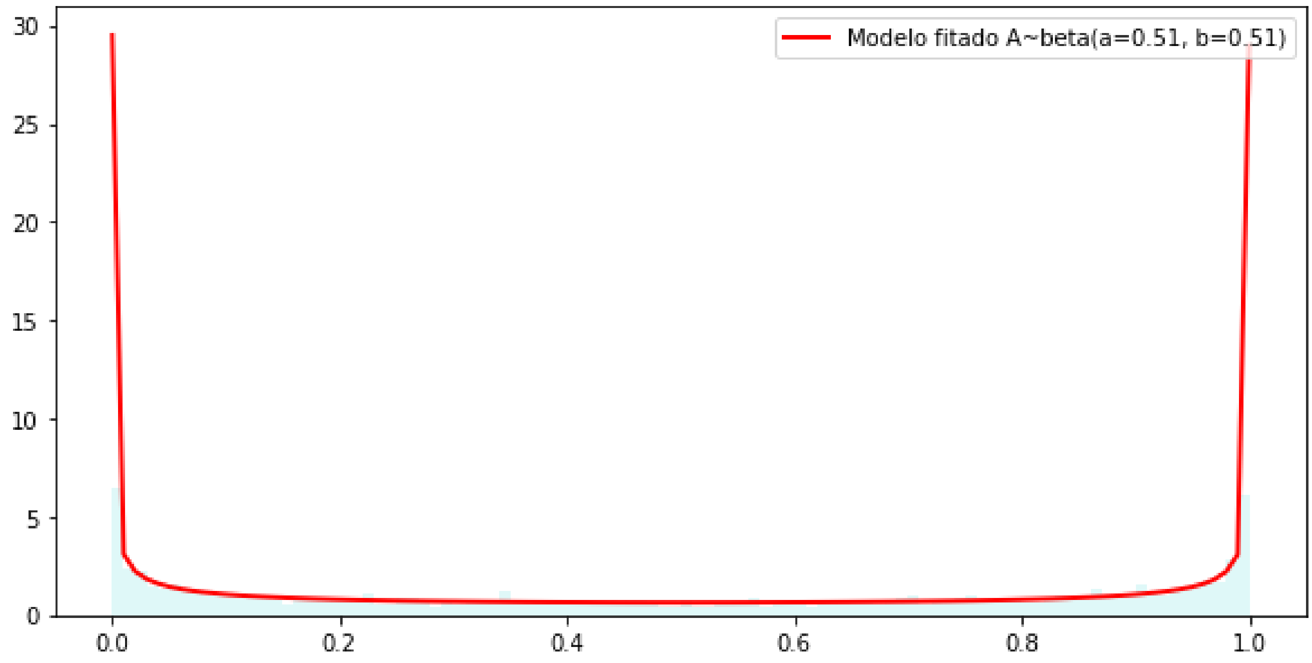
```
Out[17]:
```



2.6 Fitando melhor distribuição segundo método de Cullen-Frey

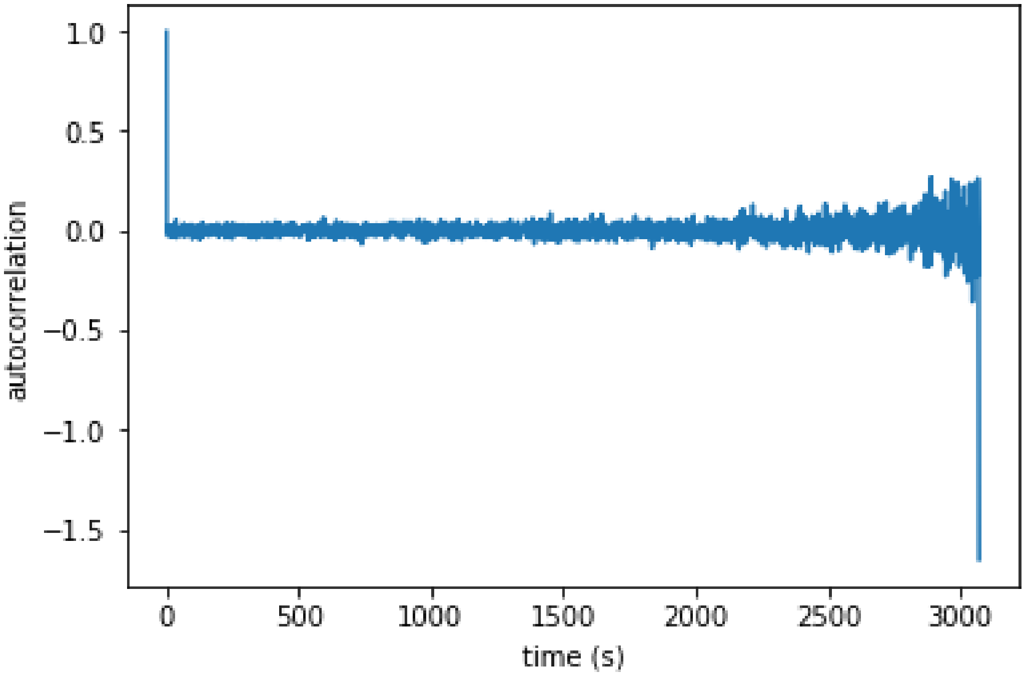
```
In [18]: aux.fitting_beta_distribution(A)
```

```
parametros de fitting:  (0.51209709972559758, 0.51410905975138699, -9.9750460756089341e-05, 1.0001996880
        Fitado                         Original
mean :  0.49901960745771895          0.50393379815
var  :  0.12343210560485554          0.1239955322
skew :  0.003688831871092711          -0.02309360954742383
kurt :  -1.490221313509967          -1.4954905588860032
```



2.7 Calculando autocorrelação

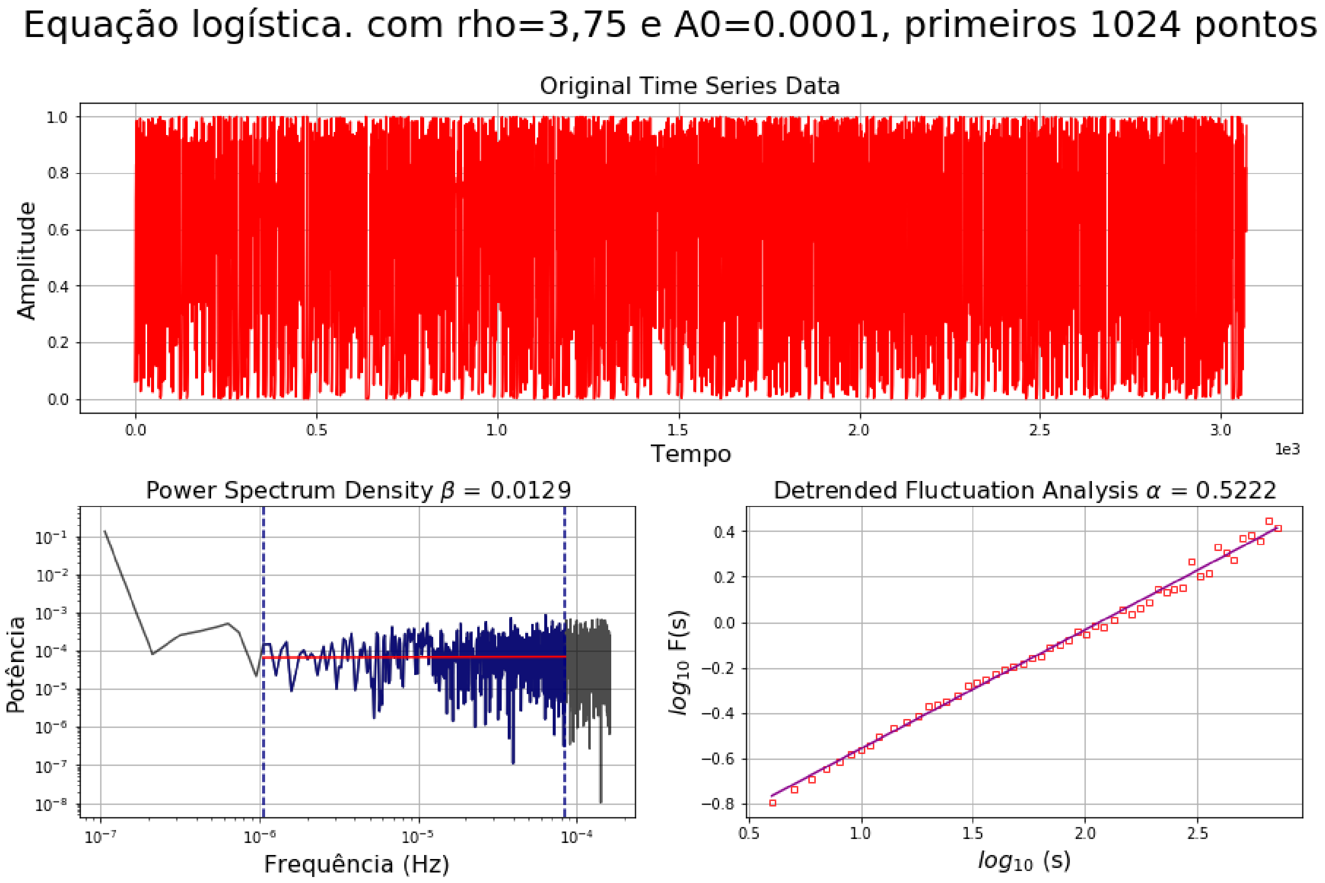
```
In [19]: aux.plot_estimated_autocorrelation(t, A, 0, len(A))
```



2.8 Plotando DFA e PSD

```
In [20]: aux.plot_psd_dfa(A, 'Equação logística. com rho=3,75 e A0=0.0001, primeiros 1024 pontos')
```

```
Original time series data (3072 points):

First 10 points: [ 0.06090836  0.22879414  0.70578953  0.83060267  0.56280749  0.98422088
  0.06212056  0.23304639  0.71494308  0.81519789]


1. Plotting time series data...
2. Plotting Power Spectrum Density...
3. Plotting Detrended Fluctuation Analysis...
```



3 Analise dos últimos 1024 pontos

```
In [21]: A = save_A[3*1024:4096]
         name = "A.ex:1.1.b"
         savetxt(name + ".txt", A)
```

```
In [22]: A.shape
```

```
Out[22]: (1024,)
```

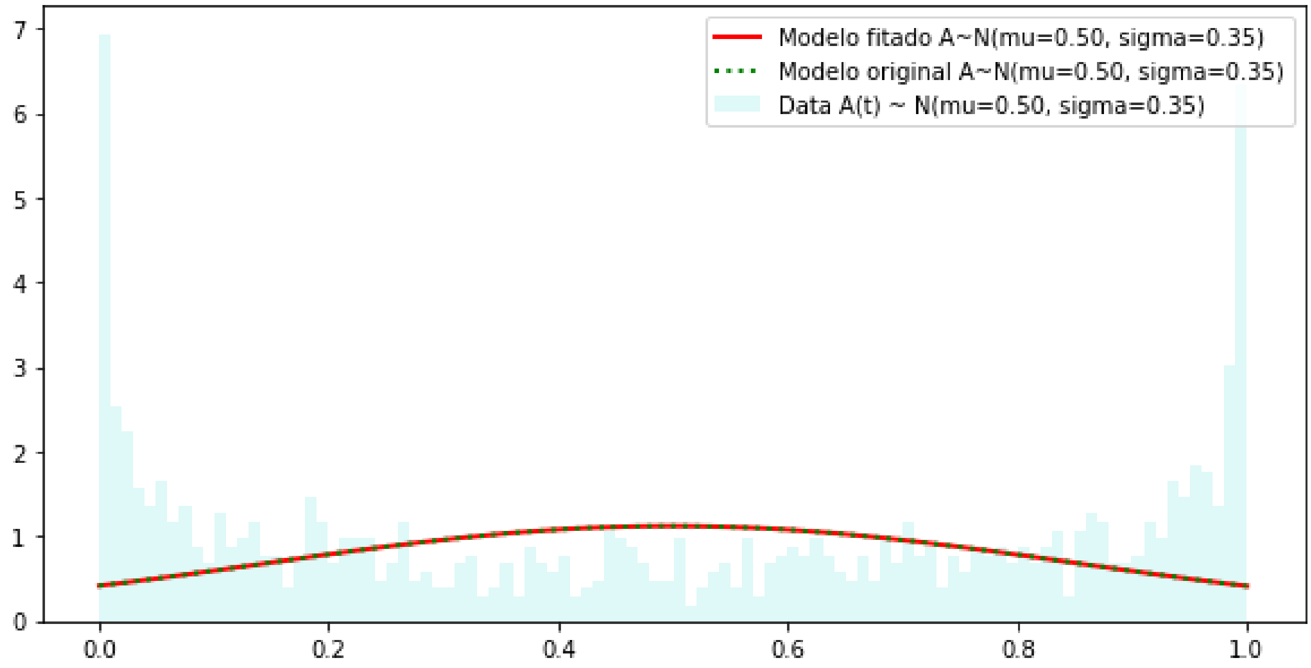
3.1 Calculando os momentos do ensemble

```
In [23]: A_mean, A_var, A_skew, A_kurtosis = aux.calcMoments(A)

         print("mean : ", A_mean)
         print("var  : ", A_var)
         print("skew : ", A_skew)
         print("kurt : ", A_kurtosis)

         A_Q1 = scoreatpercentile(A, 25)
         A_Q3 = scoreatpercentile(A, 75)

         print("Q1   : ", A_Q1)
         print("Q3   : ", A_Q3)
```

```
mean :  0.498983216203
var  :  0.125402696507
skew :  -0.00747069161779
kurt :  -1.50142291038
Q1   :  0.14436569542
Q3   :  0.852382016861
```

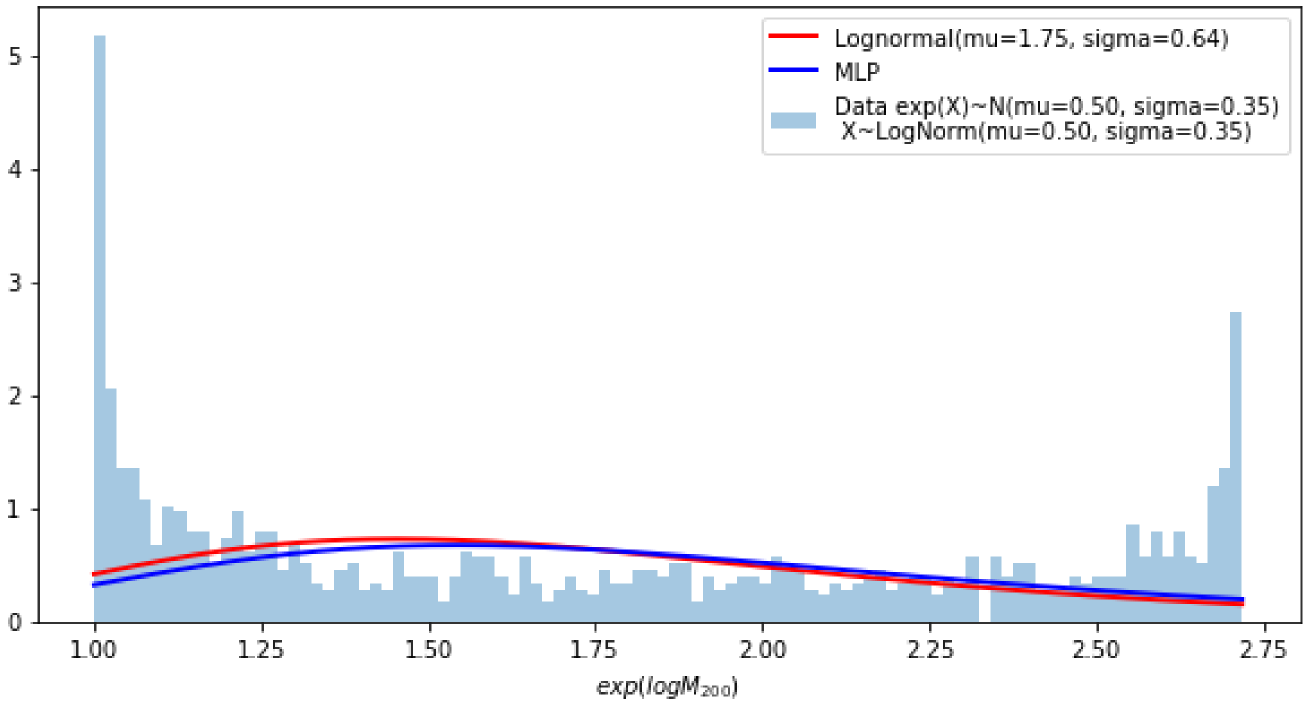
3.2 Fitando uma distribuição normal

```
In [24]: aux.fitting_normal_distribution(A)
```



3.3 Fitando uma distribuição lognormal

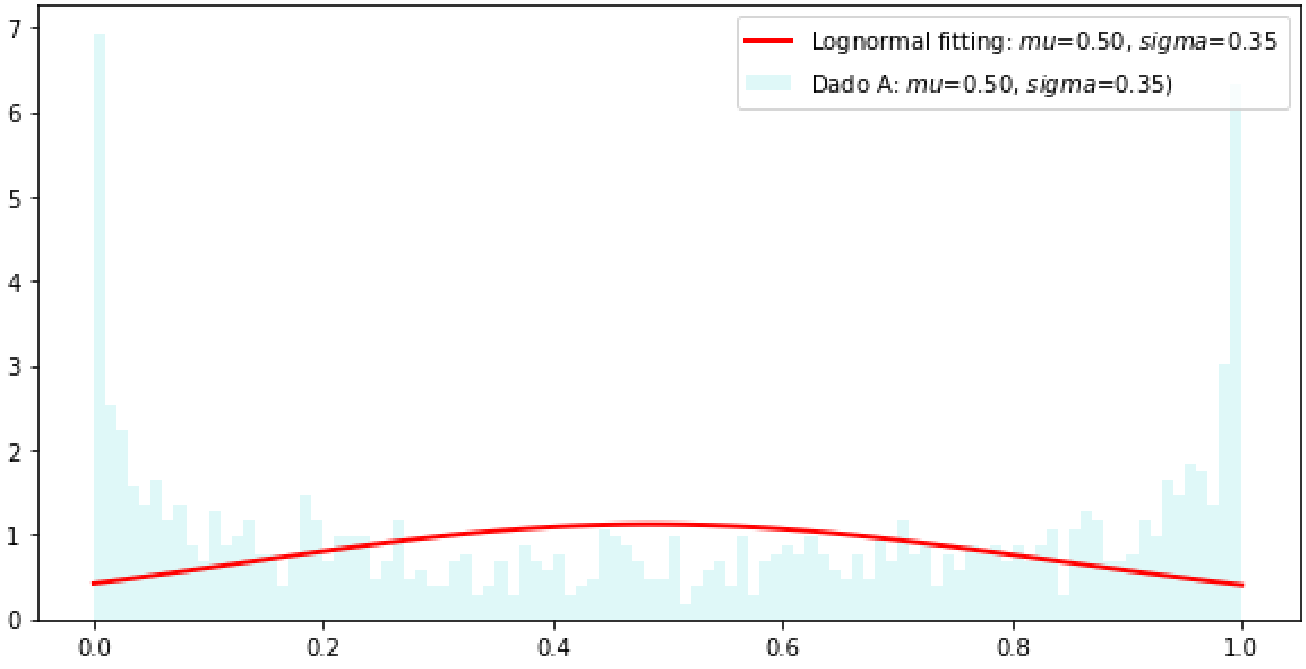
```
In [25]: aux.fitting_lognormal_and_mlp_distribution(A)
```



3.4 Fitando uma distribuição lognormal (utlizando minha implementação)

```
In [26]: aux.fitting_lognormal_distribution(A)
```

```
parametros de fitting:  (0.025987961429524864, -13.126470171988537, 13.621201885532102)
        Fitado                         Original
mean :  0.4993321940108135          0.498983216203
var  :  0.12543399712733683          0.125402696507
skew :  0.07799461032899589          -0.007470691617790162
kurt :  0.010816484518203495          -1.50142291037557
```

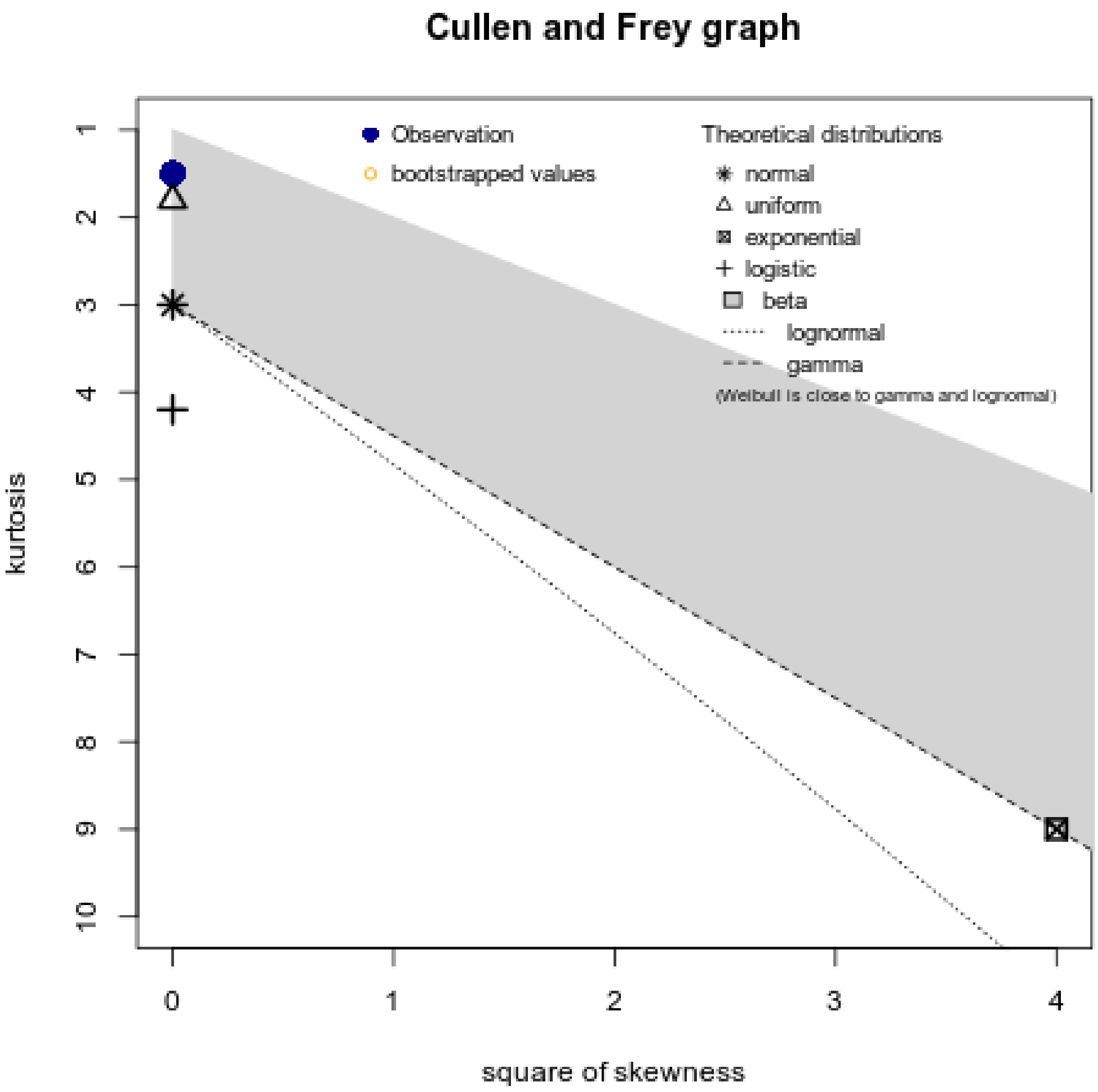


3.5 Plotando dados no espaço de Cullen-Frey

```
In [27]: command = 'Rscript'
         path_script = 'cullen_frey_script.R'

         # define arguments
         args = [name,]

         # build subprocess command
         cmd = [command, path_script] + args

         x = subprocess.check_output(cmd, universal_newlines=True)
         print(x)


         Image(name+".png")
```

```
summary statistics
------
min:  4.170881e-07   max:  0.9999999
median:  0.5012931
mean:  0.4989832
estimated sd:  0.3542955
estimated skewness:  -0.007481656
estimated kurtosis:  1.497102
```

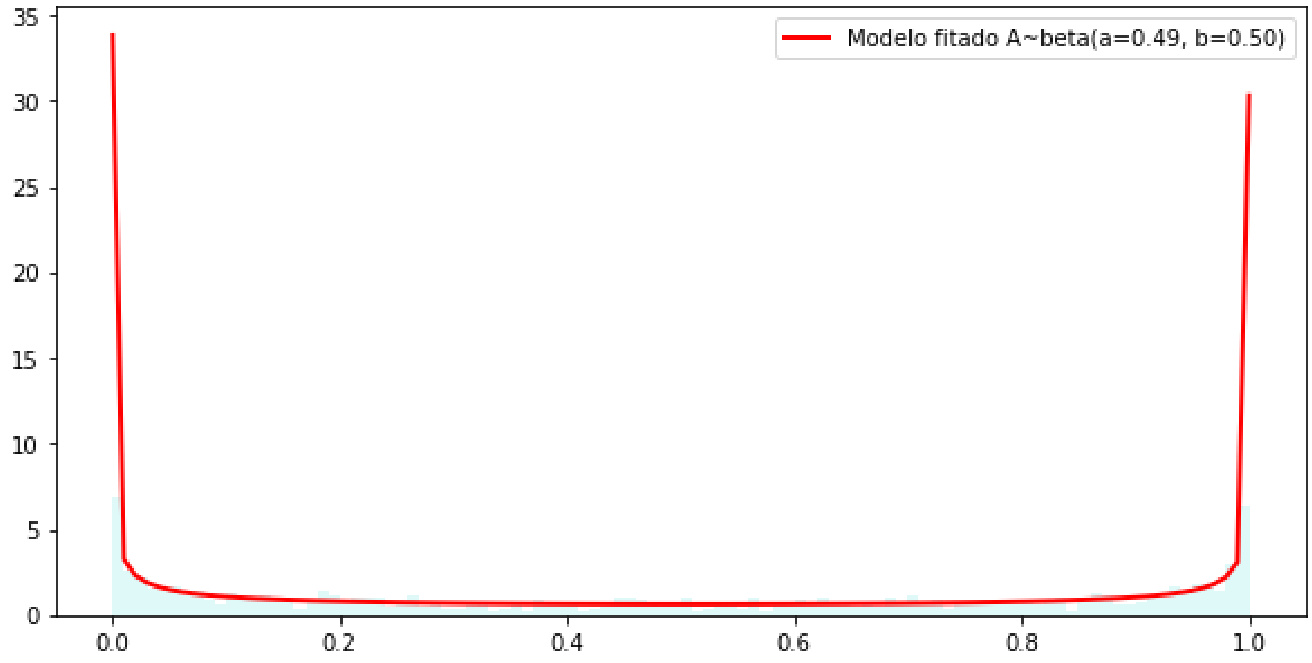
```
Out[27]:
```



3.6 Fitando melhor distribuição segundo método de Cullen-Frey

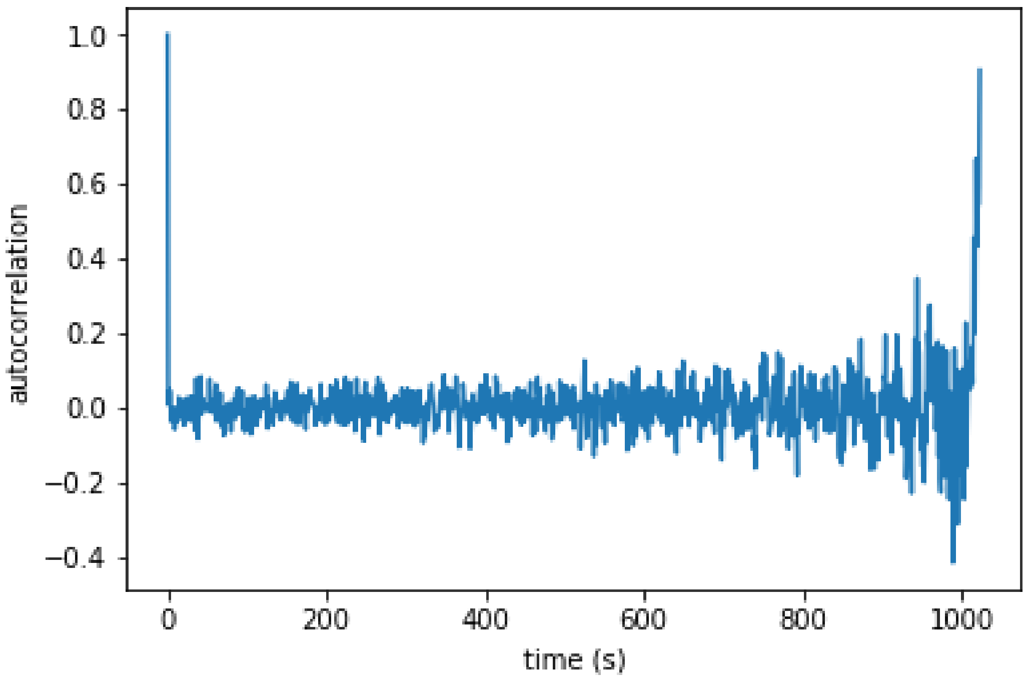
```
In [28]: aux.fitting_beta_distribution(A)
```

```
parametros de fitting:  (0.49303639436780966, 0.5049264784867592, -9.9582911899468219e-05, 1.00019947863
        Fitado                         Original
mean :  0.49404179049070096          0.498983216203
var  :  0.12515960658777517          0.125402696507
skew :  0.0224713320512054          -0.007470691617790162
kurt :  -1.5001963275568908          -1.50142291037557
```



3.7 Calculando autocorrelação

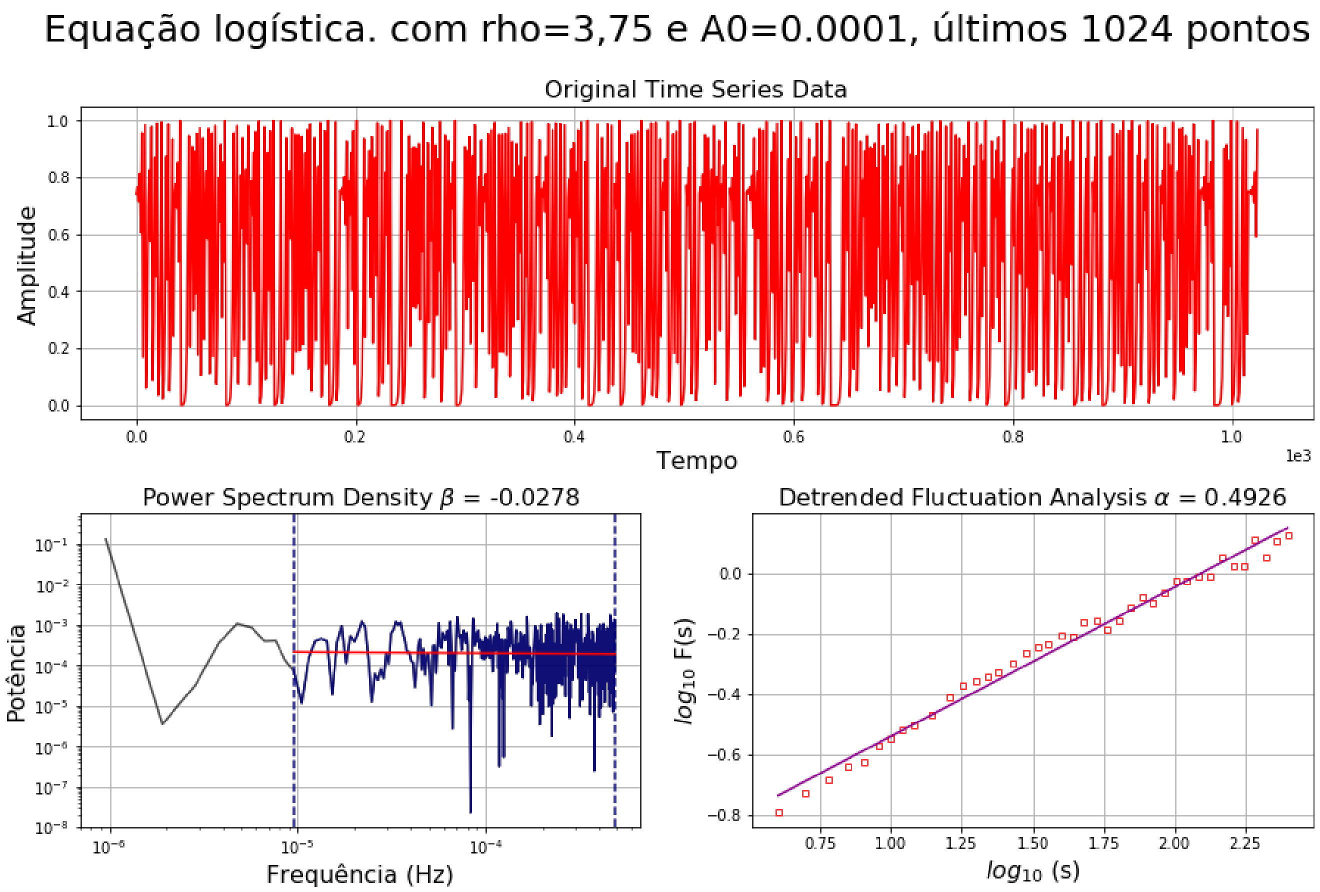
```
In [29]: aux.plot_estimated_autocorrelation(t, A, 0, len(A))
```



3.8 Plotando DFA e PSD

```
In [30]: aux.plot_psd_dfa(A, 'Equação logística. com rho=3,85 e A0=0.0001, últimos 1024 pontos')
```

```
Original time series data (1024 points):

First 10 points: [ 0.74151976  0.76667282  0.71554243  0.81416584  0.6051993   0.95573243
  0.16923182  0.56236965  0.98444011  0.06127113]


1. Plotting time series data...
2. Plotting Power Spectrum Density...
3. Plotting Detrended Fluctuation Analysis...
```


Equação logística. com rho=3,75 e A0=0.0001, últimos 1024 pontos
Original Time Series Data
Amplitude
Tempo
1e3
Power Spectrum Density β = -0.0278
Potência
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
Frequência (Hz)
Detrended Fluctuation Analysis α = 0.4926
log_{10} F(s)
log_{10} (s)