Ruído Browniano

27 de maio de 2018

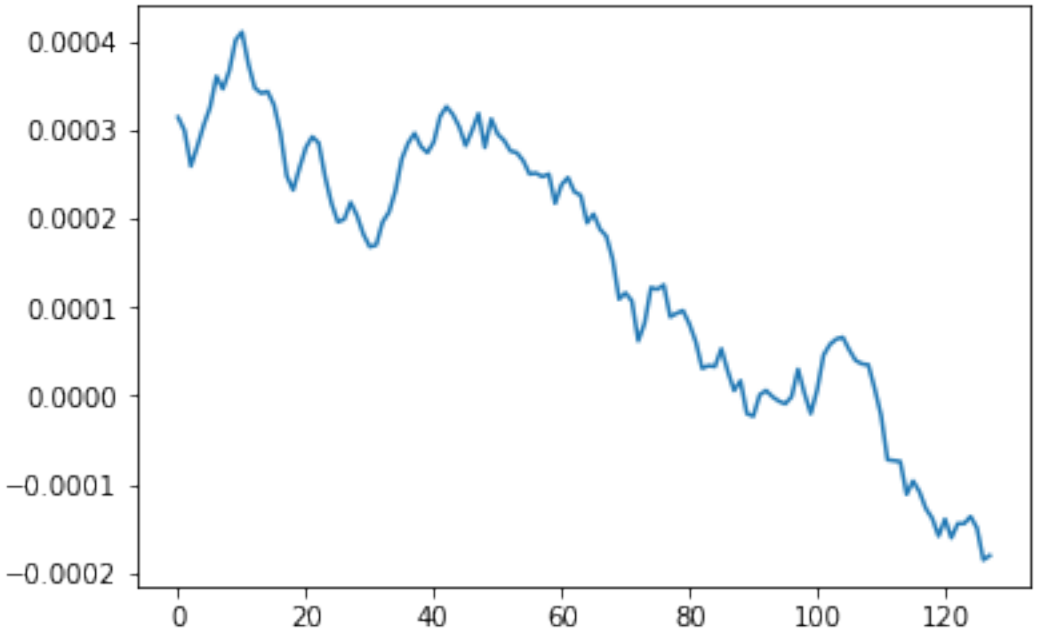
In [1]:
```
from scipy.stats import moment
from scipy.stats import kurtosis, skew, scoreatpercentile
from scipy.stats import norm, lognorm, beta
from scipy.optimize import minimize

from numpy import zeros, fromiter, savetxt, loadtxt
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:1.3.d"

A = loadtxt("noise_equals_2.txt")

savetxt(name + ".txt", A)

save_A = A
```

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



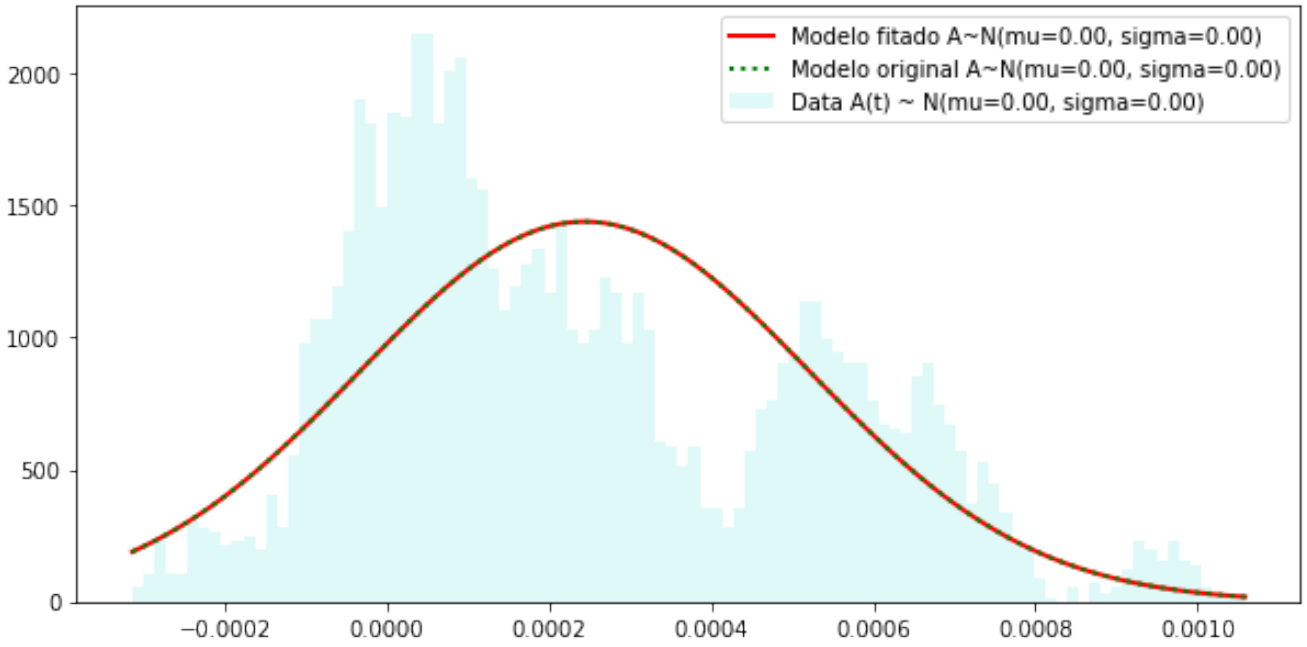
1.2 Calculando os momentos do ensemble

```
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)
```

```
mean :  0.000244137451172
var  :  7.68228788123e-08
skew :  0.552554516864
kurt :  -0.526957500598
```

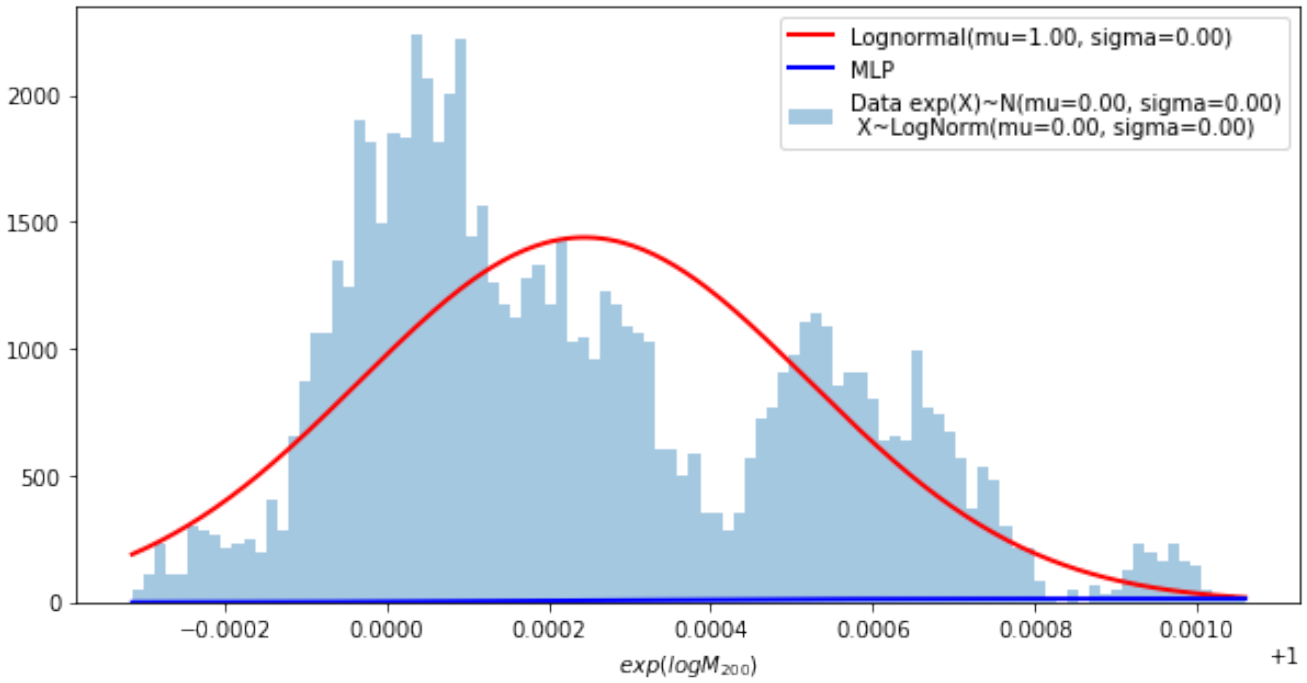
1.3 Fitando uma distribuição normal

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



1.4 Fitando uma distribuição lognormal

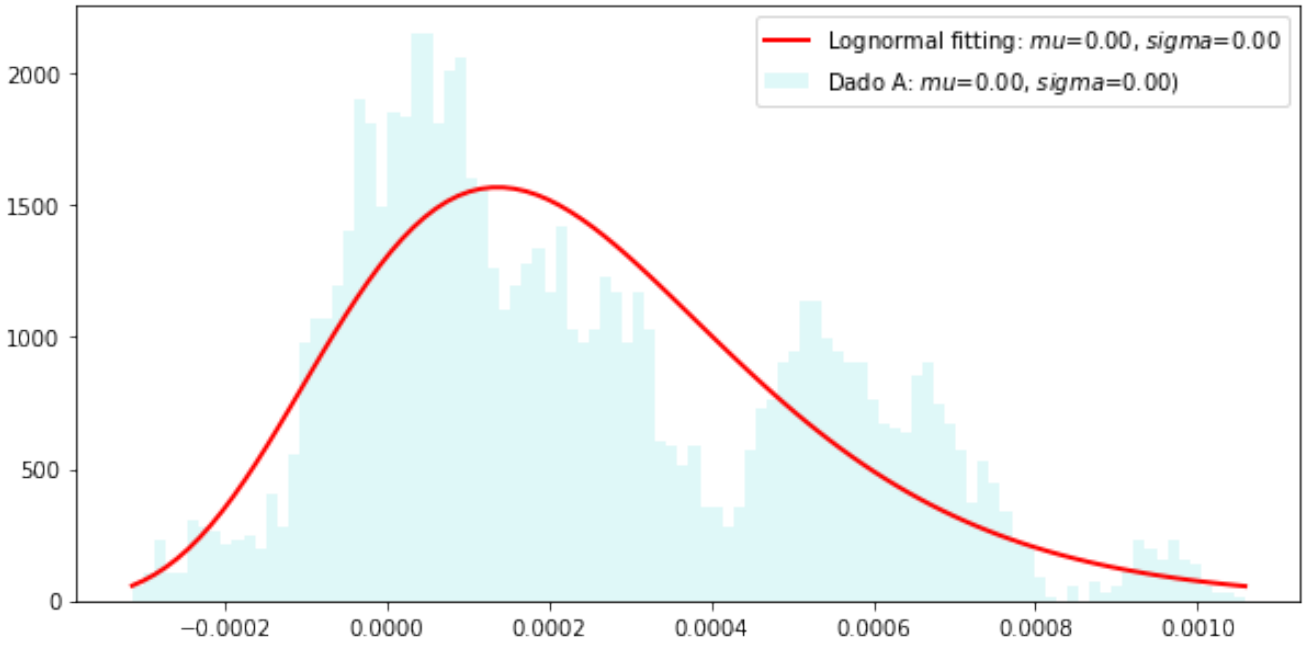
In [6]: `aux.fitting_lognormal_and_mlp_distribution(A)`



1.5 Fitando uma distribuição lognormal (utlizando minha implementação)

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

```
parametros de fitting:  (0.27572050283128752, -0.00075154414848649723, 0.00095853280766207839)
        Fitado                          Original
mean :  0.00024412466715190866           0.000244137451172
var  :  7.830336620534376e-08           7.68228788123e-08
skew :  0.8653325455765708          0.5525545168644652
kurt :  1.3603550915502023          -0.5269575005976916
```

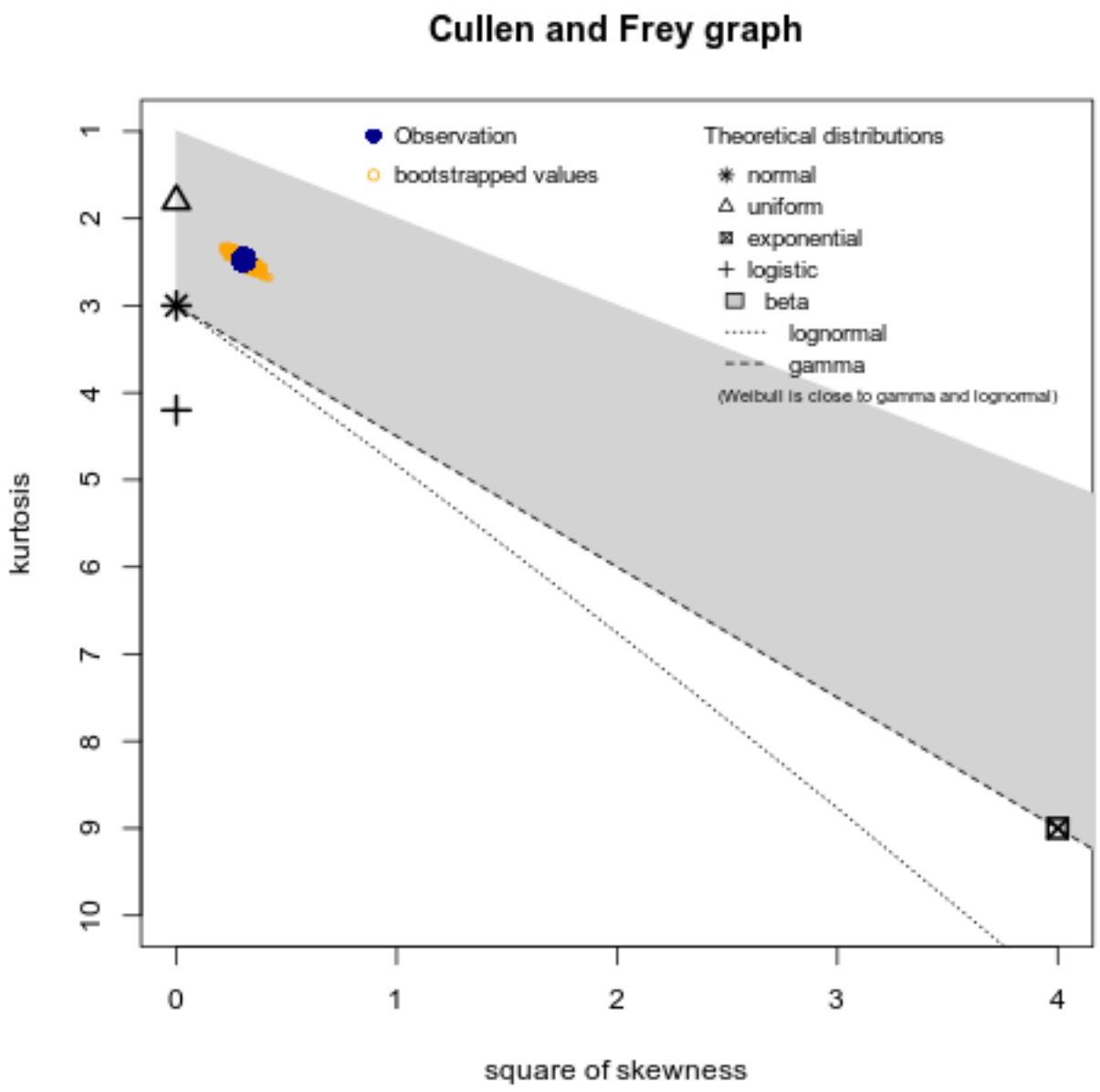


1.6 Plotando dados no espaço de Cullen-Frey

```
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:  -0.000314   max:  0.001059
median:  0.0001815
mean:  0.0002441375
estimated sd:  0.0002772032
estimated skewness:  0.552757
estimated kurtosis:  2.473865
```

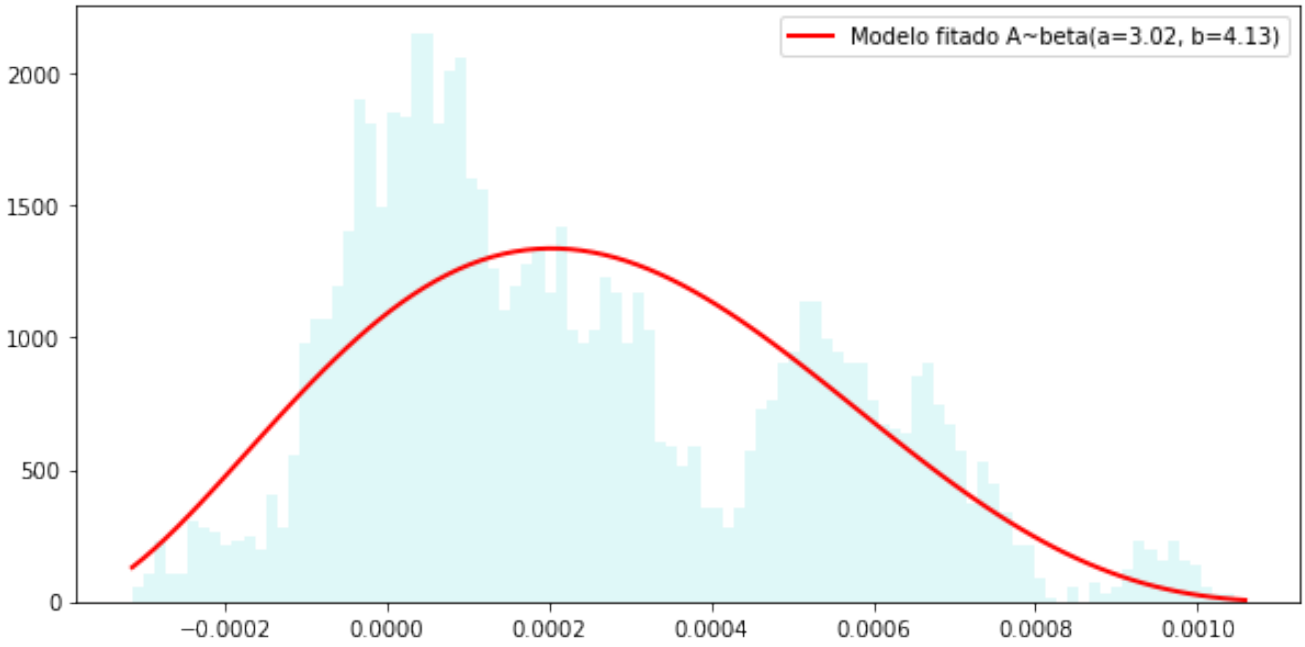
Out[8]:



1.7 Fitando melhor distribuição segundo método de Cullen-Frey

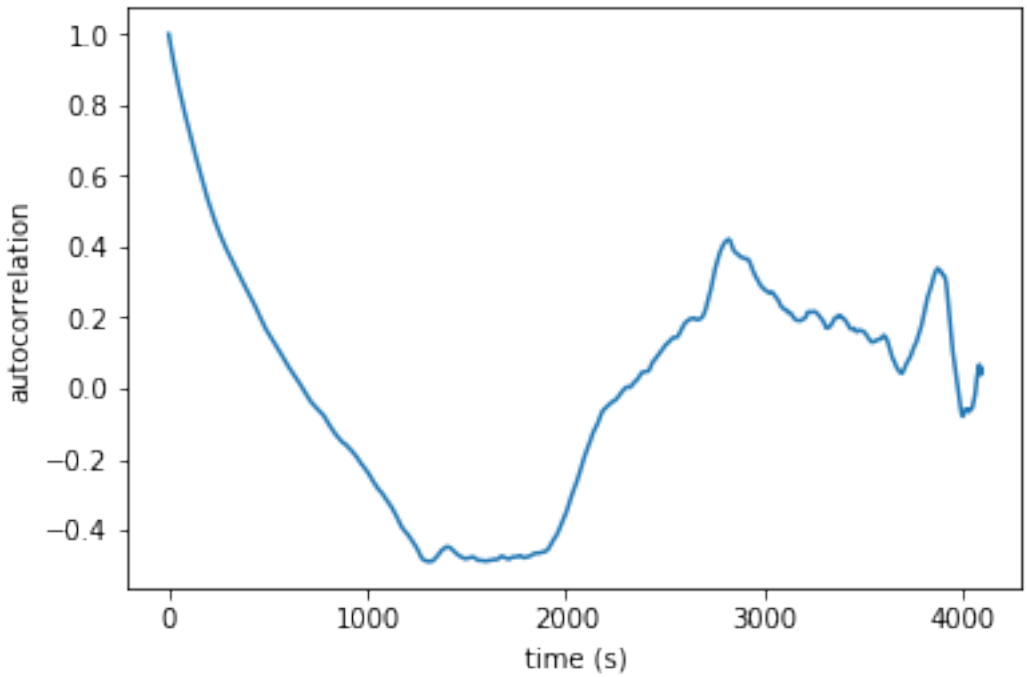
In [9]: `aux.fitting_beta_distribution(A)`

```
parametros de fitting:  (3.0229491910543635, 4.1328353893033425, -0.00041399999999999998, 0.001573000000
        Fitado                          Original
mean :  0.00025051121105308834           0.000244137451172
var  :  7.402120291715554e-08           7.68228788123e-08
skew :  0.19588729808864974           0.5525545168644652
kurt :  -0.5389060385985043           -0.5269575005976916
```



1.8 Calculando autocorrelação

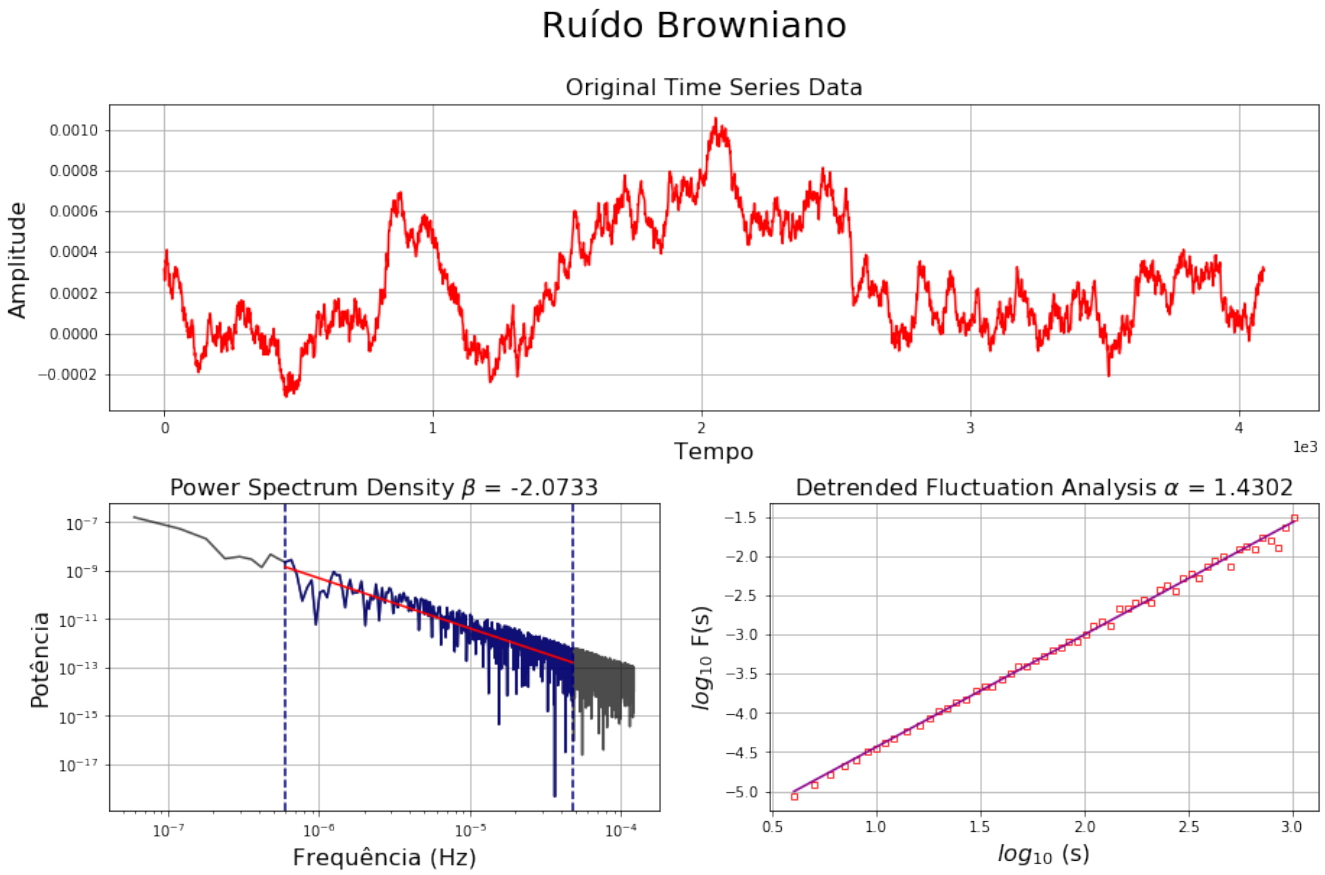
```
In [10]: aux.plot_estimated_autocorrelation(t, A, 0, len(A))
```



1.9 Plotando DFA e PSD

```
In [11]: aux.plot_psd_dfa(A, 'Ruído Browniano')
```

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

First 10 points: [ 0.000314  0.000299  0.000259  0.000281  0.000305  0.000325  0.00036
  0.000346  0.000366  0.000401]


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:1.3.d"
         savetxt(name + ".txt", A)
```

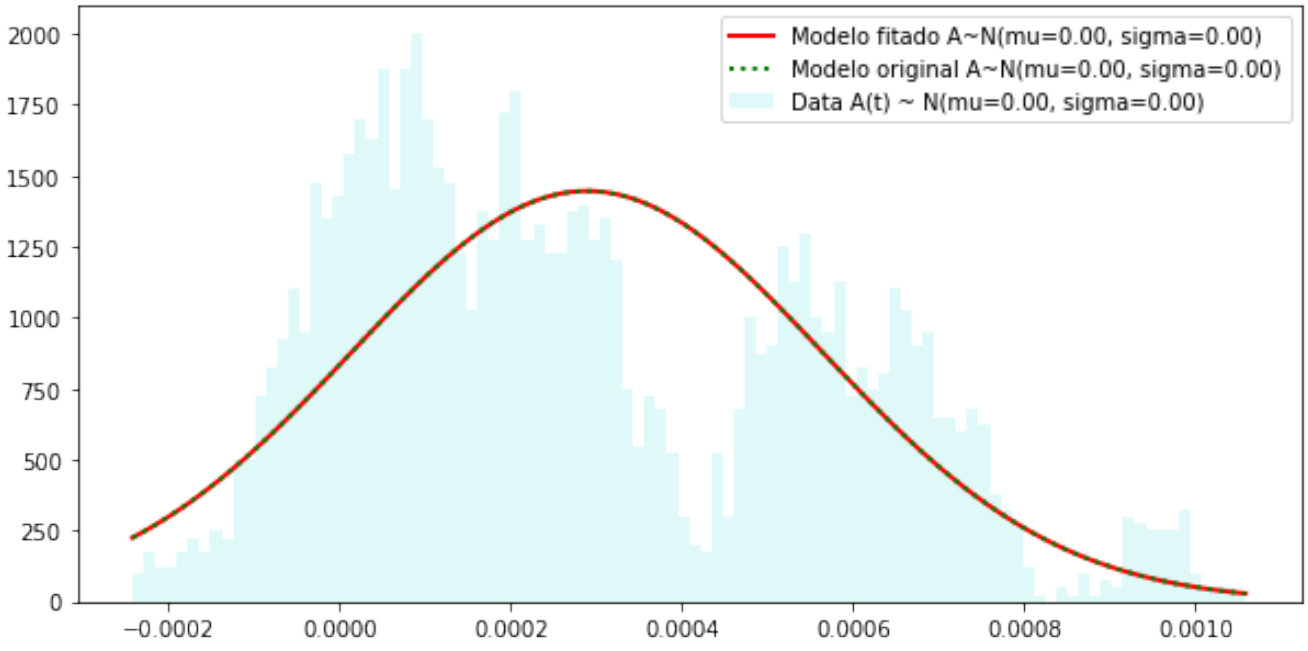
2.1 Calculando os momentos do ensemble

```
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)
```

```
mean :  0.000289416015625
var  :  7.60334102643e-08
skew :  0.492223906348
kurt :  -0.656631729673
```

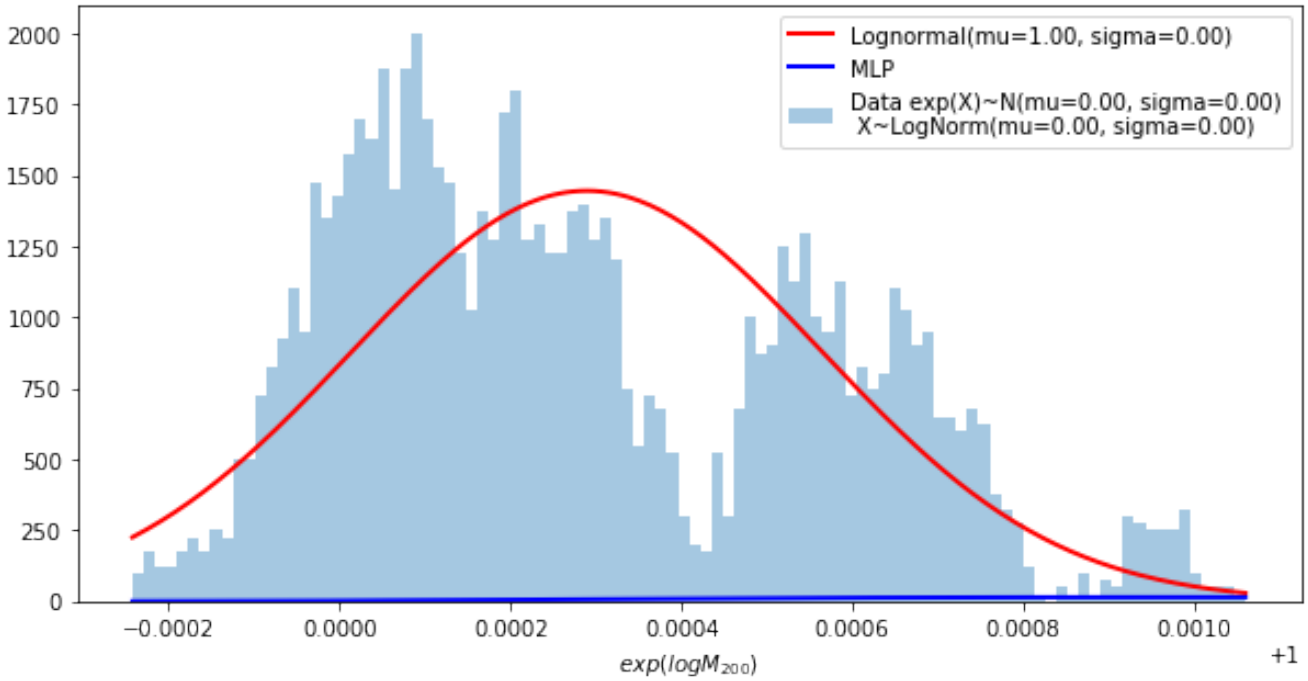
2.2 Fitando uma distribuição normal

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



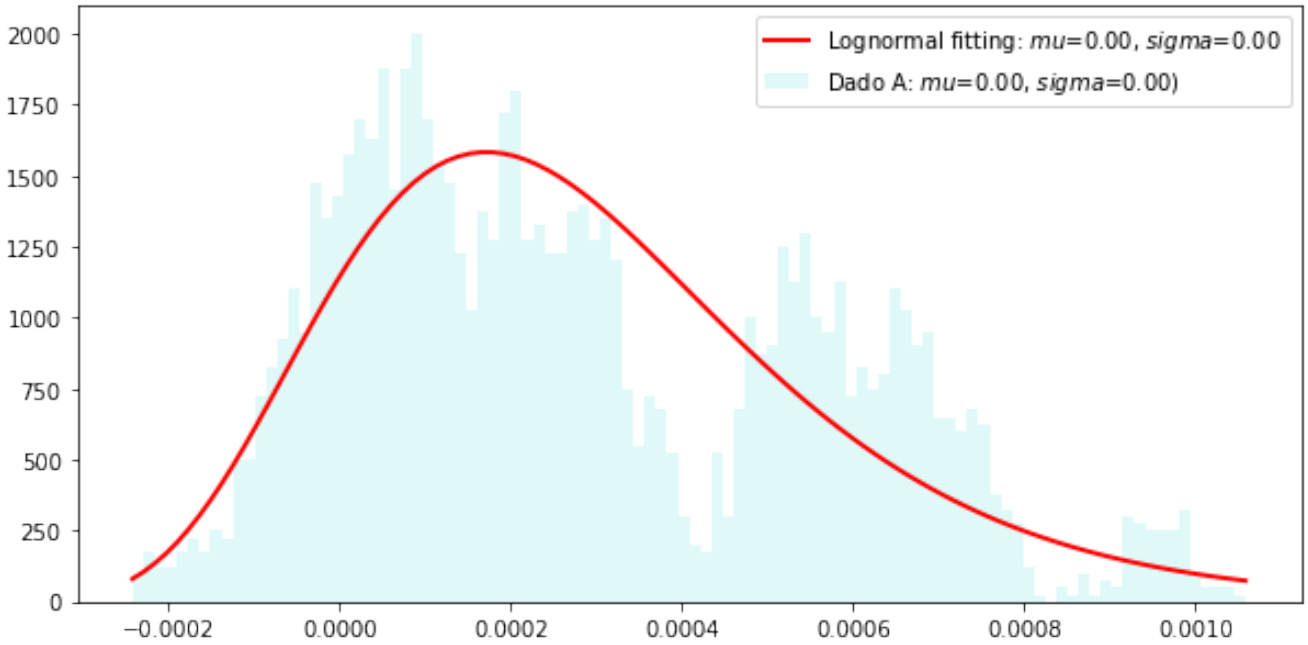
2.3 Fitando uma distribuição lognormal

```
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:  (0.30462333992505852, -0.0006176151541133289 4, 0.0008661985212019829 3)
        Fitado                          Original
mean :  0.00028971991823909156           0.000289416015625
var  :  8.005119479313776e-08           7.60334102643e-08
skew :  0.9658077292738554           0.49222390634796254
kurt :  1.703226903562415           -0.6566317296729003
```

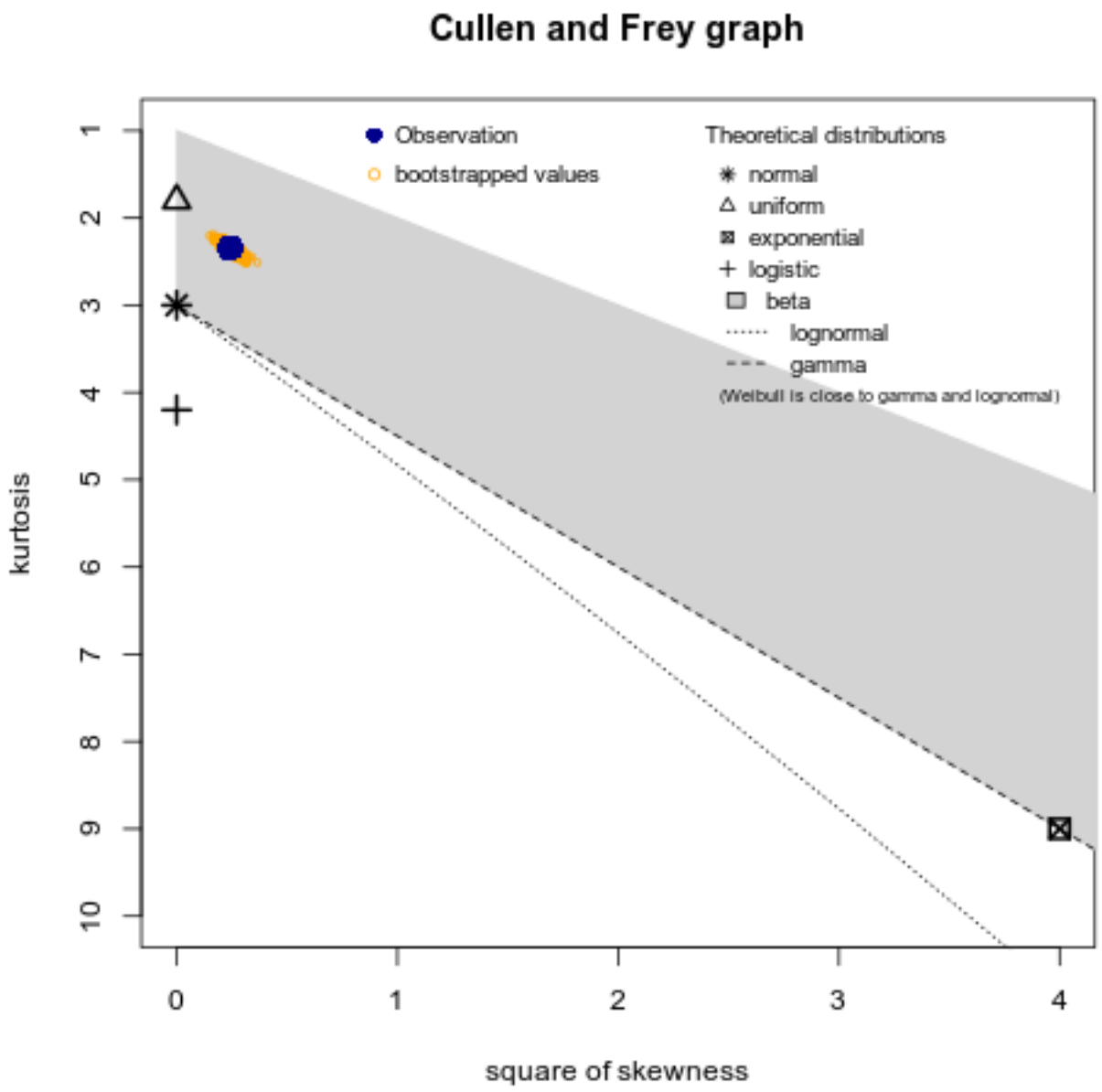


2.5 Plotando dados no espaço de Cullen-Frey

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:  -0.000242   max:  0.001059
median:  0.000231
mean:  0.000289416
estimated sd:  0.0002757865
estimated skewness:  0.4924644
estimated kurtosis:  2.344254
```

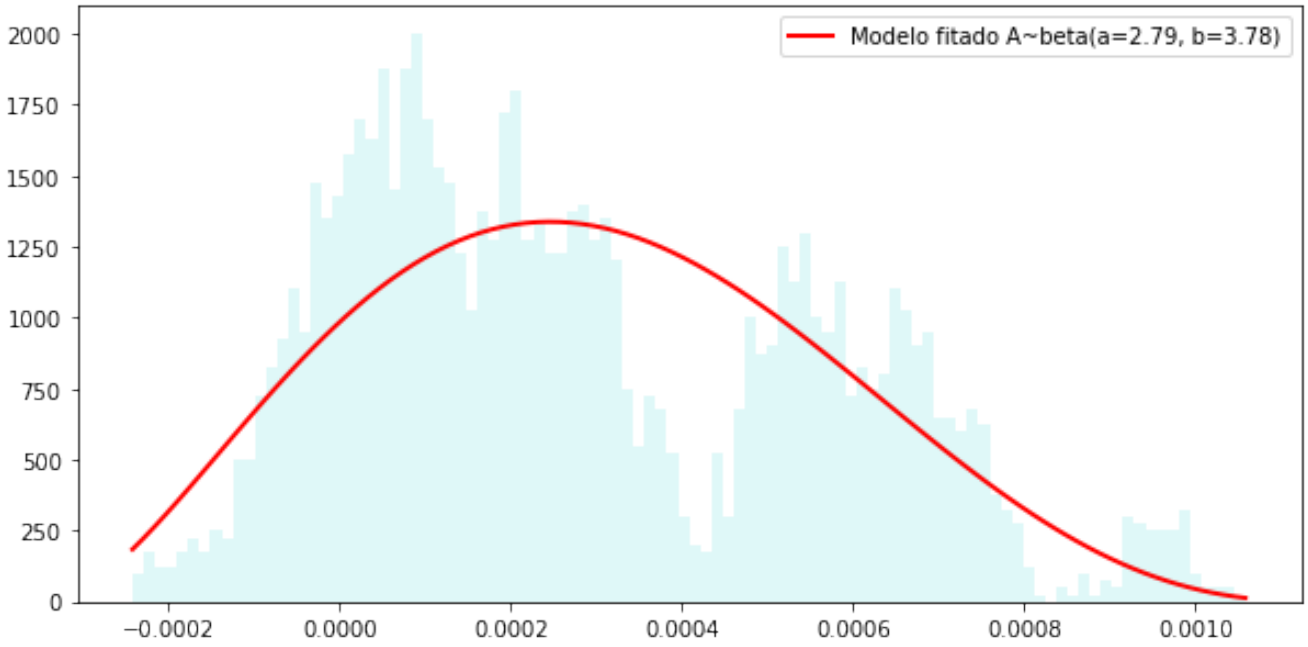
Out[17]:



2.6 Fitando melhor distribuição segundo método de Cullen-Frey

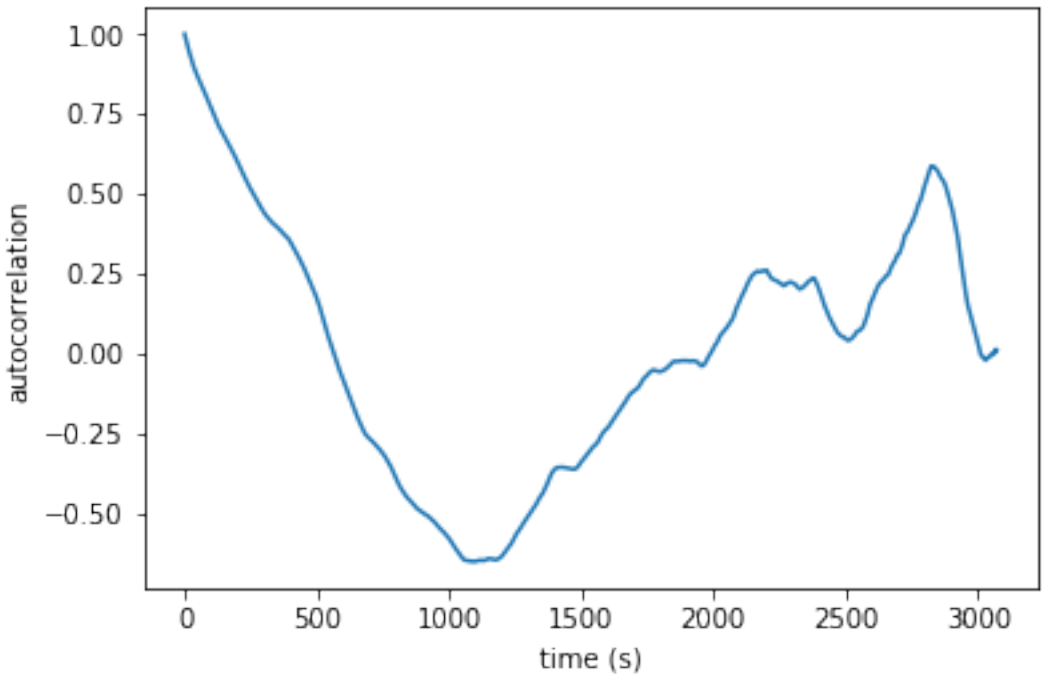
```
In [18]: aux.fitting_beta_distribution(A)
```

```
parametros de fitting:  (2.7881563991506124, 3.7760423395852598, -0.00034200000000000002, 0.001501000000
        Fitado                          Original
mean :  0.00029555271918092746           0.000289416015625
var  :  7.277613674222103e-08            7.60334102643e-08
skew :  0.19554852093283057             0.49222390634796254
kurt :  -0.5759779443165691             -0.6566317296729003
```



2.7 Calculando autocorrelação

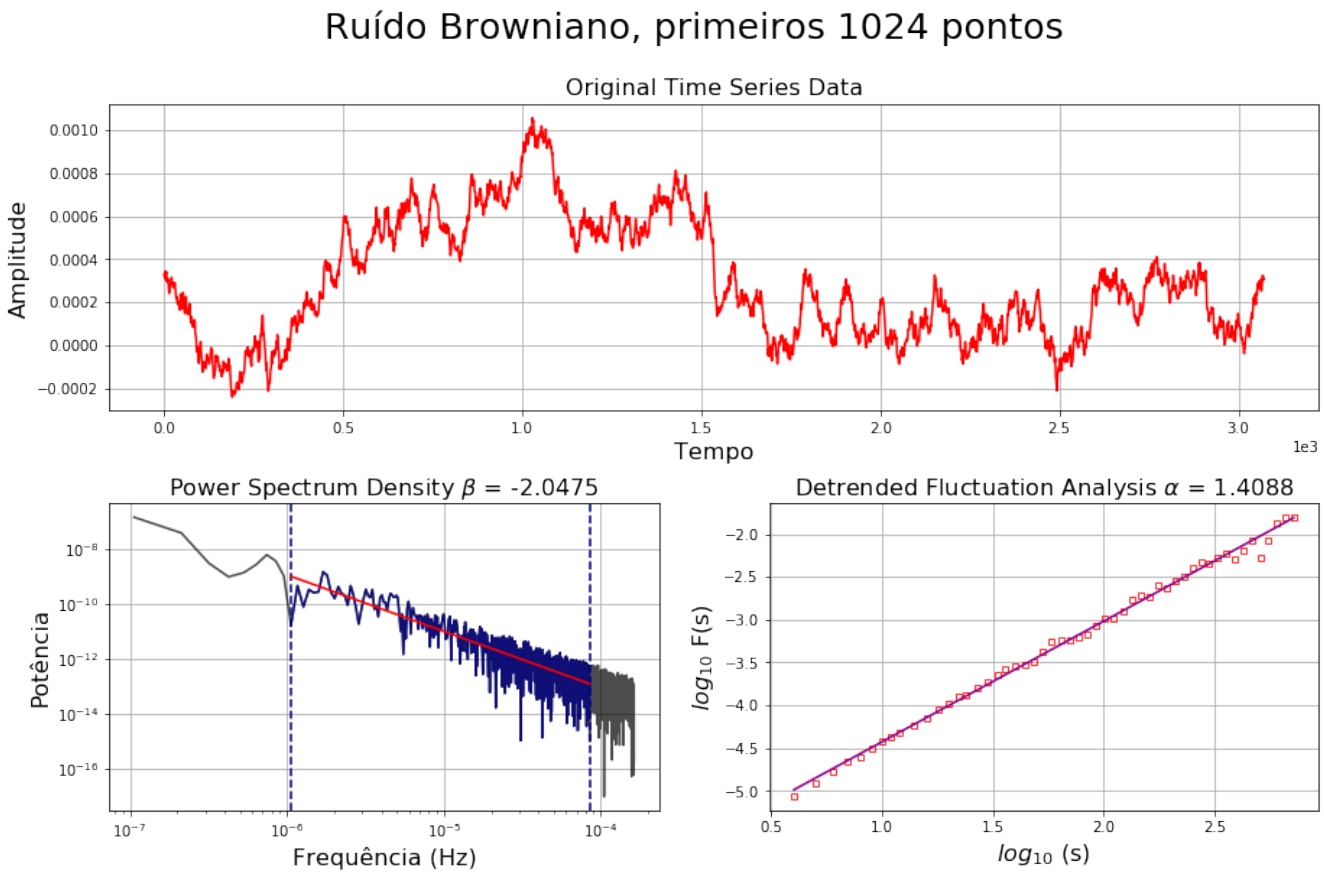
```
In [19]: aux.plot_estimated_autocorrelation(t, A, 0, len(A))
```



2.8 Plotando DFA e PSD

```
In [20]: aux.plot_psd_dfa(A, 'Ruído Browniano, primeiros 1024 pontos')
```

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

First 10 points: [ 0.000329  0.000318  0.00031   0.000343  0.000343  0.000331  0.000341
  0.000313  0.000295  0.000311]


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:1.3.d"
         savetxt(name + ".txt", A)

In [22]: A.shape

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

3.1 Calculando os momentos do ensemble

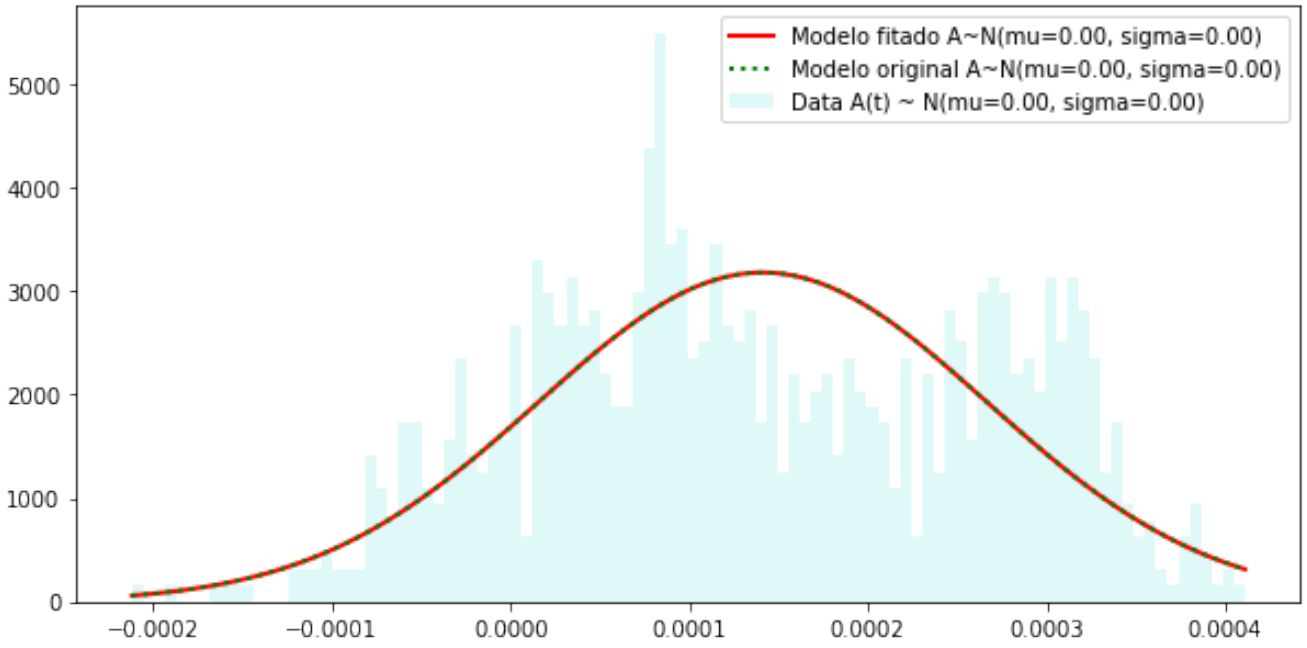
```
In [23]: A_mean, A_var, A_skew, A_kurtosis = aux.calcMoments(A)

         print("mean : ", A_mean)
```

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

```
mean :  0.000141081054688
var  :  1.57486057348e-08
skew :  0.00889858728318
kurt :  -0.905696021218
```

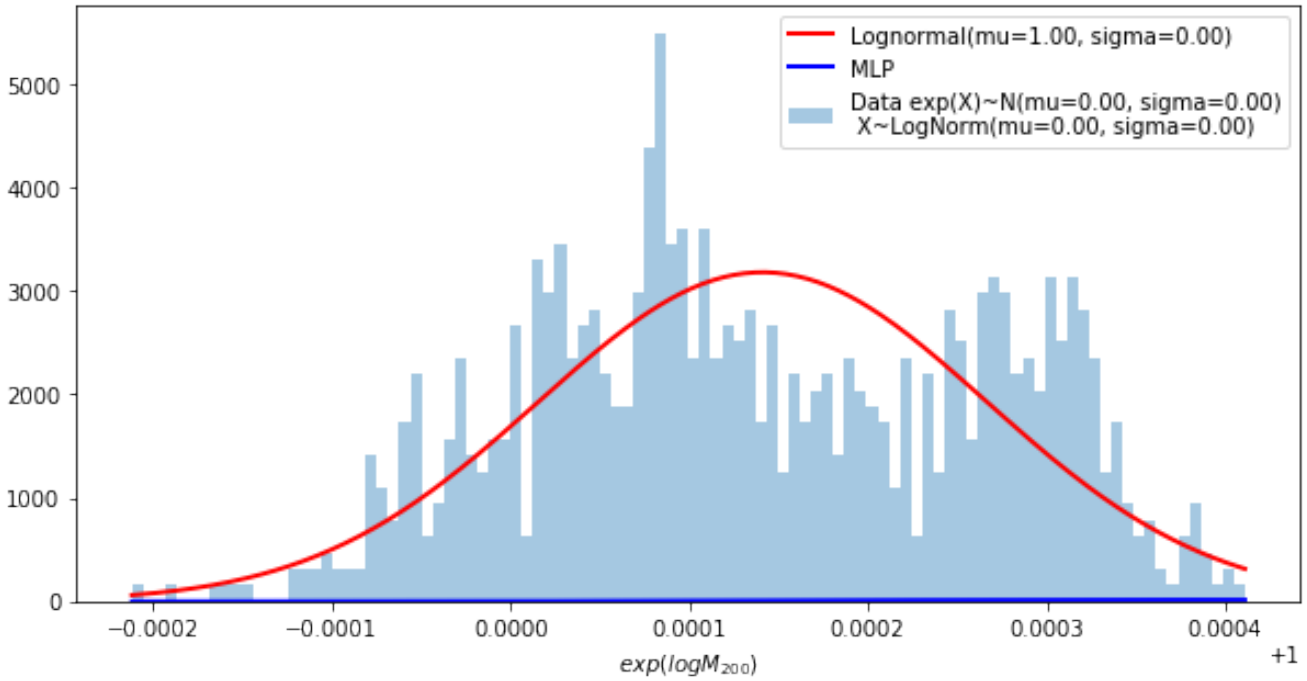
3.2 Fitando uma distribuição normal

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



3.3 Fitando uma distribuição lognormal

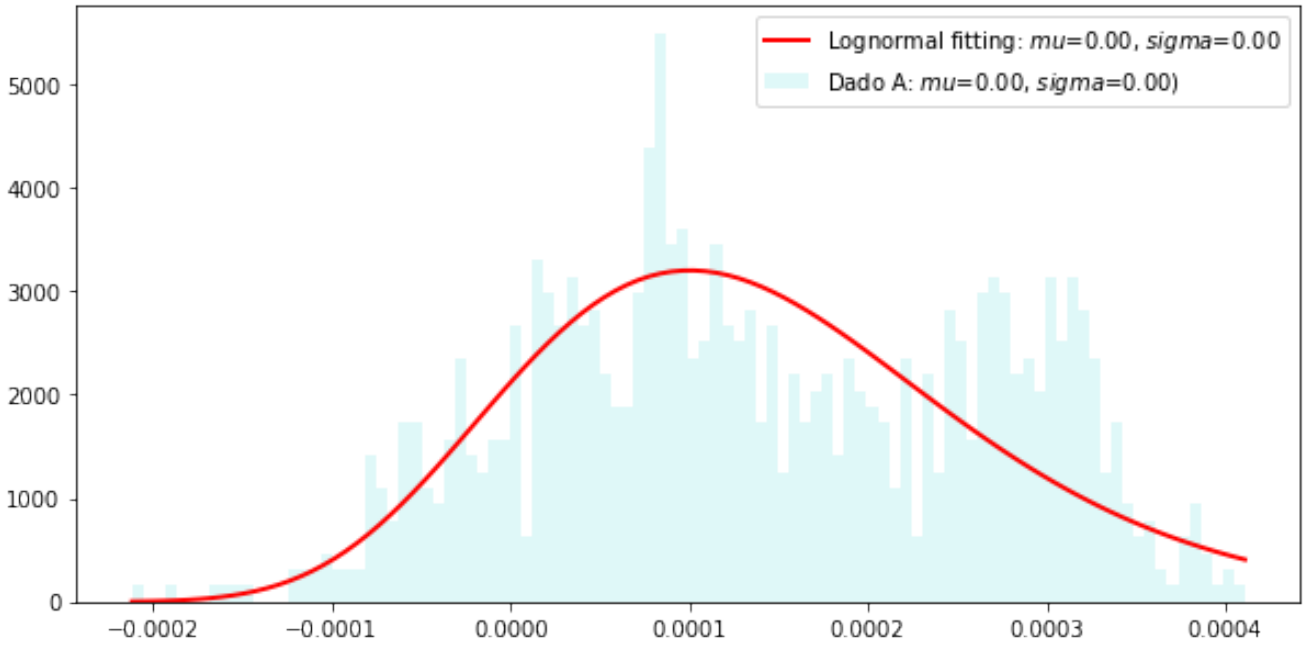
In [25]: `aux.fitting_lognormal_and_mlp_distribution(A)`



3.4 Fitando uma distribuição lognormal (utlizando minha implementação)

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

```
parametros de fitting:  (0.21743451661782692, -0.00045984127487144457, 0.00058727444037858358)
        Fitado                          Original
mean :  0.00014148106232931304           0.000141081054688
var  :  1.7505653642840885e-08           1.57486057348e-08
skew :  0.6707422453651557          0.008898587283178017
kurt :  0.8104548422392481          -0.9056960212181342
```

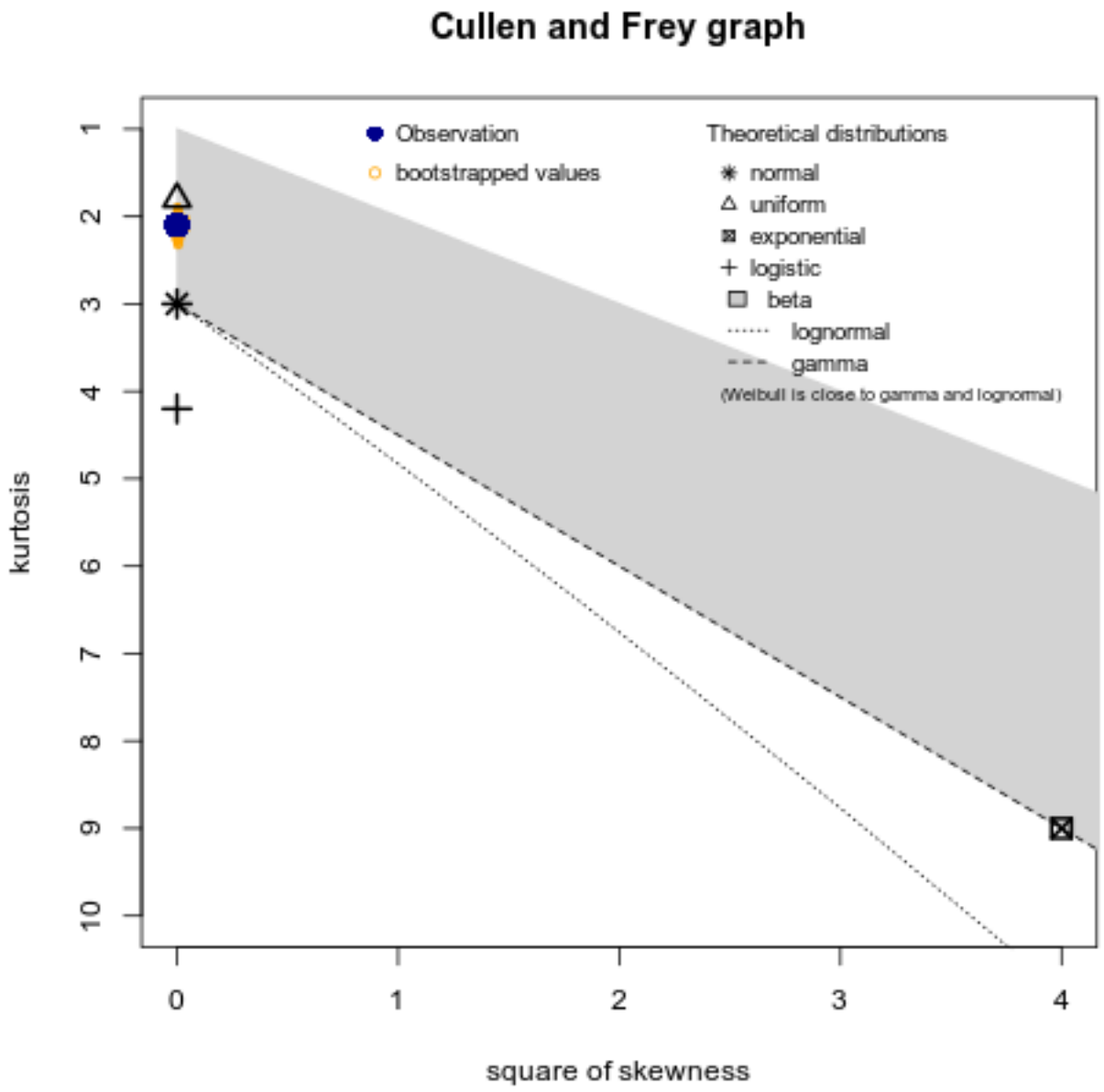


3.5 Plotando dados no espaço de Cullen-Frey

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:  -0.000212   max:  0.000411
median:  0.000127
mean:  0.0001410811
estimated sd:  0.0001255548
estimated skewness:  0.008911647
estimated kurtosis:  2.095748
```

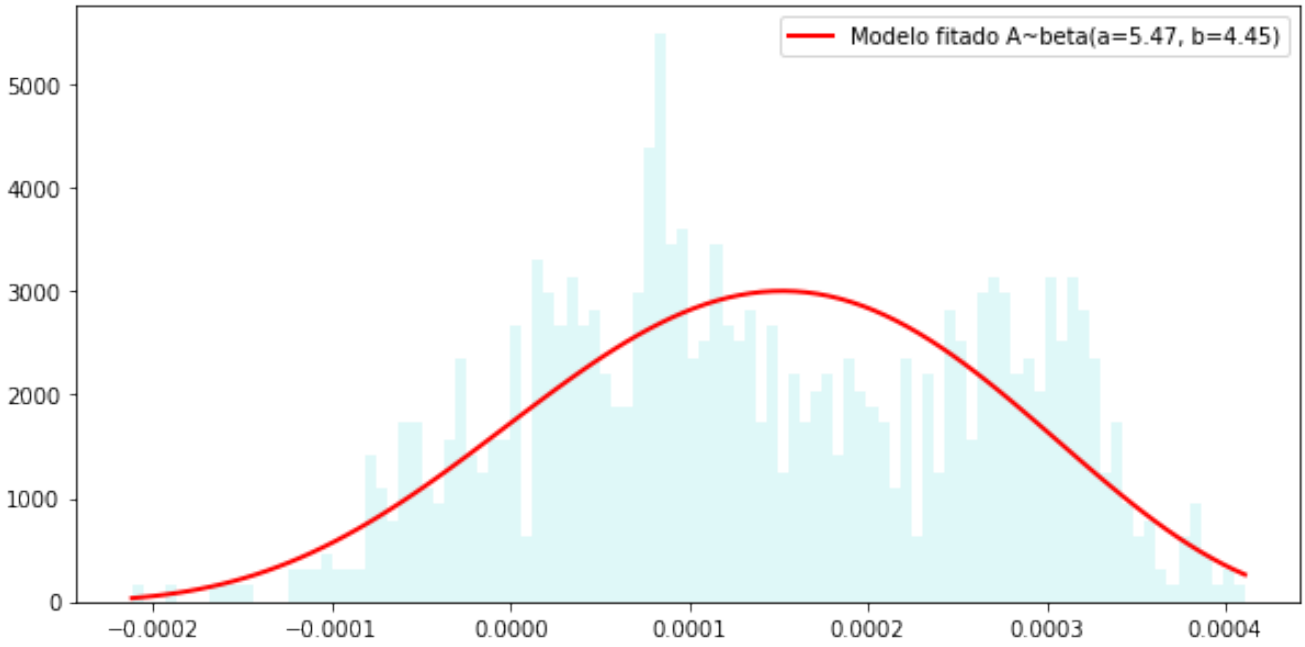
Out[27]:



3.6 Fitando melhor distribuição segundo método de Cullen-Frey

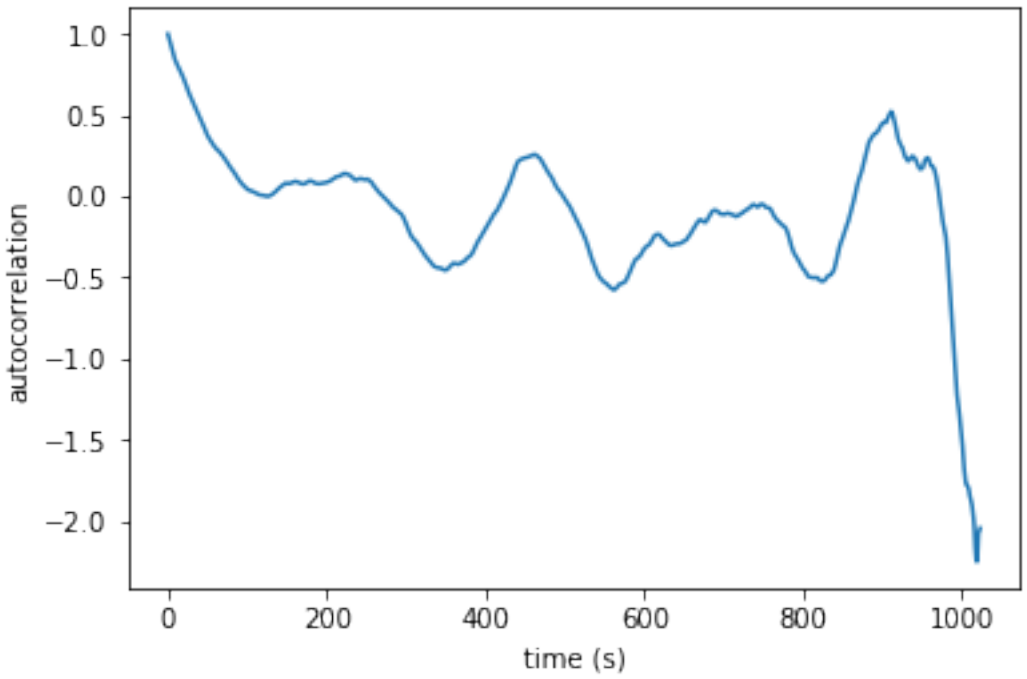
```
In [28]: aux.fitting_beta_distribution(A)
```

```
parametros de fitting:  (5.4657271200783075, 4.4547102501875706, -0.00031199999999999999, 0.000823000000
        Fitado                          Original
mean :  0.0001414370060444108           0.000141081054688
var  :  1.534494744508124e-08           1.57486057348e-08
skew :  -0.11360109578211047           0.008898587283178017
kurt :  -0.446520983143211            -0.9056960212181342
```



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, 'Ruído Browniano, últimos 1024 pontos')
```

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

First 10 points: [ -5.30000000e-05  -5.10000000e-05  -5.90000000e-05  -8.10000000e-05
  -8.70000000e-05  -8.80000000e-05  -6.90000000e-05  -4.40000000e-05
  -2.50000000e-05  -2.60000000e-05]


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