



MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES  
**INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS**

## **LISTA DE EXERCÍCIOS CAP-239B**

Leonardo Sattler Cassara

Lista de Exercícios apresentada ao  
Prof. Reinaldo R. Rosa como parte  
da avaliação do curso CAP-239.

Arquivos se encontram em:  
[github/Lista-CAP-239B](https://github.com/lsattler/CAP-239B)

INPE  
São José dos Campos  
12 de junho de 2020

# PREFÁCIO

Os códigos desta lista estão organizados da seguinte maneira:

- pasta **signal\_generator\_codes**: contém os scripts geradores dos sinais para cada série da tabela *Dataset\_signal*. São eles:
  - noise.py: gerador randômico não-gaussiano. Inputs são:  $n$  (tamanho da série) e  $res$  (resolução).
  - colornoise.py: gerador de ruídos coloridos. Inputs são:  $exponent$  (expoente do espectro de potência) e  $size$  (tamanho da série).
  - pmnoise.py: gerador de série temporal via p-model. Inputs são:  $noValues$  (tamanho da série) e  $p$  (parâmetro do p-model).
  - chaosnoise.py: gerador de duas séries caóticas, que são o mapa Logístico e o mapa de Henon. Para o Logístico, inputs são:  $N$  (tamanho da série),  $\rho$  e  $\tau$  (parâmetros da equação logística). Para o mapeamento de Henon, os inputs são:  $N$  (tamanho da série),  $a$  e  $b$  (parâmetros da série).
- pasta **statistical\_analysis\_codes**: contém todos os scripts para as análises estatísticas desta lista. São eles:
  - stats\_tools.py: script com diversas funções relevantes para a lista. Contém função para normalização, determinação de variância, skewness e kurtosis além dos quatro momentos estatísticos.
  - CullenFrey\_R\_Python.py: script que classifica a série no espaço de Cullen-Frey. Ele foi implementado a partir de um ambiente R dentro do Python com o módulo rpy2 (do Python) e o pacote fitdistrplus (do R). Inputs são: *input\_data* (dados de input), *boot* (booleano representando opção de bootstrap) e *name* (prefixo dos arquivos salvos).
  - Distribution\_Fitter\_R.py: script de ajuste de distribuições que inclui benchmark das PDF's ajustadas (salvo em formato .txt). Implementado a partir de um ambiente R dentro do Python via módulo rpy2 (do Python) e dos pacotes fitdistrplus, ismev e actuar (do R). É possível também ajustar uma GEV aos dados. Inputs são: *input\_data* (série temporal), *fit\_dist* (lista com nomes das distribuições a serem ajustadas à série), *fit\_GEV* (booleano que representa se GEV será

ajustada ou não), *method* (método de ajuste) e *name* (prefixo dos arquivos salvos).

- Distribution\_Fitter\_Python.py: script para ajuste de distribuições com benchmarks das PDF's ajustadas (salvo em formato .txt). Implementado com o pacote fitter do Python. É possível também ajustar uma GEV aos dados. Inputs são: *input\_data* (série temporal), *fit\_dist* (lista com nomes das distribuições a serem ajustadas à série), *fit\_GEV* (booleano que representa se GEV será ajustada ou não) e *name* (prefixo dos arquivos salvos).
- kmeans\_2D.py: script para análise k-means num espaço de parâmetros bidimensional. Ele gera plots dos dados no espaço sendo analisado e determina o melhor número de clusters pelo método do cotovelo. Gera um arquivo kmeans.csv contendo informação das coordenadas x e y do centróide e da inércia para cada número de clusters testado. Inputs são: *df* (dataframe bidimensional), *n\_c* (número de clusters a serem testados), *axis\_labels* (lista com strings para label dos eixos) e *name* (prefixo dos arquivos salvos).
- kmeans\_2D\_group\_flags.py: o mesmo que o kmeans\_2D.py, porém com a separação dos inputs por agrupamento em arquivos .csv separados pela label de cada cluster identificado. Input extra: *flags* (lista de strings com identificadores dos pontos dentro do dataframe de input *df*).
- kmeans\_3D.py: o mesmo que o kmeans\_2D.py, porém para um espaço de parâmetros tridimensional.
- kmeans\_3D\_plus\_data.py: o mesmo que o kmeans\_3D.py, porém com plot de dados extras no espaço de parâmetros sendo considerado. Os inputs extras são: *plus\_data* (lista de coordenadas tridimensionais dos dados a serem plotados) e *data\_labels* (lista de strings para cada ponto extra a ser plotado).
- Specplus.py: script para gerar o PSD (Power Spectral Density) e o DFA (Detrended Fluctuation Analysis) de uma série temporal. Inputs são: *data* (dados da série) e *fig\_name* (prefixo dos arquivos salvos).
- mfdfa\_ss\_m1.py, mfdfa\_ss\_m2.py, mfdfa\_ss\_m3.py e mfdfa\_ss\_m4.py: quatro módulos que executam a MFDFA (Multifractal Detrended Fluctuation Analysis). O script que implementa a análise é o mfdfa\_ss\_m4.py, com os seguintes inputs: *dx* (série temporal),

*plot\_bool* (booleano para permitir ou não plot dos resultados) e *name* (prefixo dos arquivos salvos).

- soc.py: script para cálculo de Self Organized Criticality. O inputs é *data*, os dados da série a ser analisada.

Além destes, os scripts para solução dos exercícios estão em diretórios separados (por questão, da primeira à nona) dentro da pasta **Exercises**. Eles importam os códigos acima apresentados, de forma que as pastas (em negrito) devem existir e estar referenciadas corretamente na importação.

Por último, as séries temporais *ST-Sol3GHz* e *ST-surftemp504* se encontram na pasta **time\_series\_data**. As séries relativas ao número de casos da Covid-19, *ST-OWS\_NDC\_Covid19*, são baixadas automaticamente nas questões em que sua análise é requisitada, de forma a obter os valores mais atualizados.

## SUMÁRIO

	<u>Pág.</u>
<b>Exercício 1 . . . . .</b>	<b>1</b>
<b>Exercício 2 . . . . .</b>	<b>3</b>
<b>Exercício 3 . . . . .</b>	<b>5</b>
<b>Exercício 4 . . . . .</b>	<b>8</b>
4.1 . . . . .	8
4.2 . . . . .	10
<b>Exercício 5 . . . . .</b>	<b>13</b>
5.1 . . . . .	13
5.2 . . . . .	19
<b>Exercício 6 . . . . .</b>	<b>20</b>
6.1 . . . . .	20
6.2 . . . . .	23
6.3 . . . . .	25
<b>Exercício 7 . . . . .</b>	<b>27</b>
7.1 . . . . .	27
7.2 . . . . .	27
7.3 . . . . .	31
<b>Exercício 8 . . . . .</b>	<b>32</b>
8.1 . . . . .	32
8.2 . . . . .	33
<b>Exercício 9 . . . . .</b>	<b>34</b>
9.1 . . . . .	34
9.2 . . . . .	35

## Exercício 1 - Simulação de Sinais Estocásticos com GRNG1.py com N valores de medidas

Os resultados das análises dos sinais referentes a este exercício se encontram na pasta **Exercise1**. Abaixo está um plot com dez sinal para a série com  $N = 64$  da família noise, junto com seus respectivos histogramas.

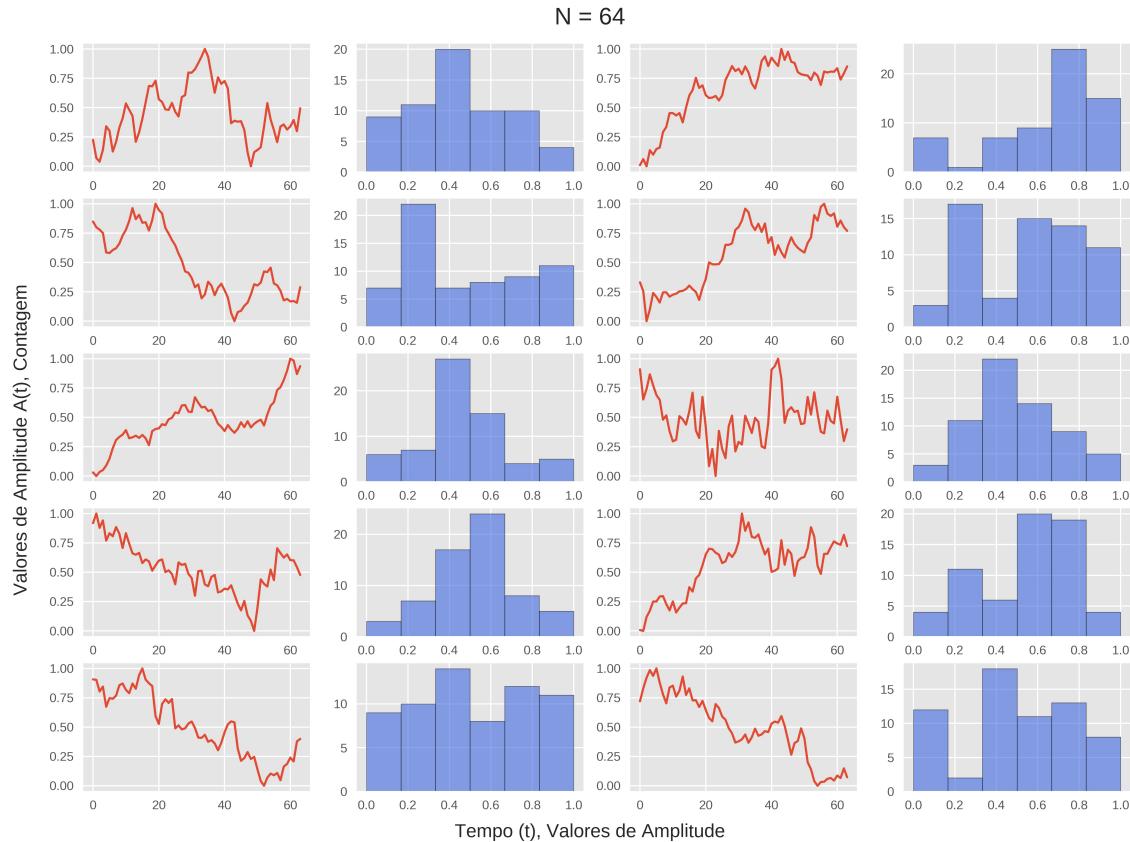


Figura 1.1: Dez sinal e seus respectivos histogramas para a série com  $N = 64$  do grupo noise.

Os resultados da análise foram armazenados em três arquivos .csv: *Exercicio1\_momentos.csv*, contendo os quatro momentos estatísticos de cada sinal gerado, *Exercicio1\_parametros.csv*, com a variância, skewness e kurtosis de todos os sinal, e *Exercicio1\_kmeans.csv*, que armazena o resultado da análise dos clusters via kmeans, com as coordenadas de cada centróide gerado para cada número de cluster testado e suas respectivas inércias. Foram testados de um a oito clusters para as séries da família noise. Os plots a seguir ilustram o resultado da aplicação do algoritmo *kmeans\_3D.py* no espaço de parâmetros variância x skewness x kurtosis. O método do cotovelo identificou que três clusters são ideais para classificar os sinal

desta família.

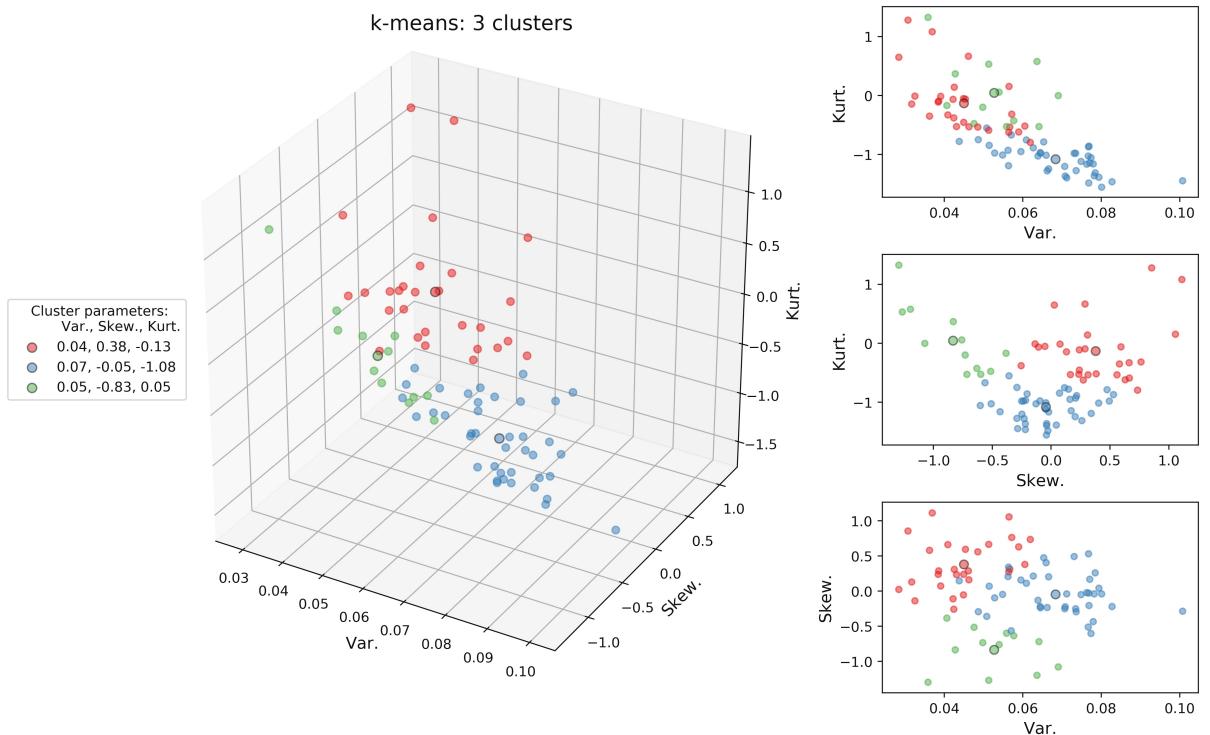


Figura 1.2: Técnica kmeans no espaço de parâmetros variância x skewness x kurtosis para a família de sinais noise. Resultado para número de clusters  $n\_c = 3$ .

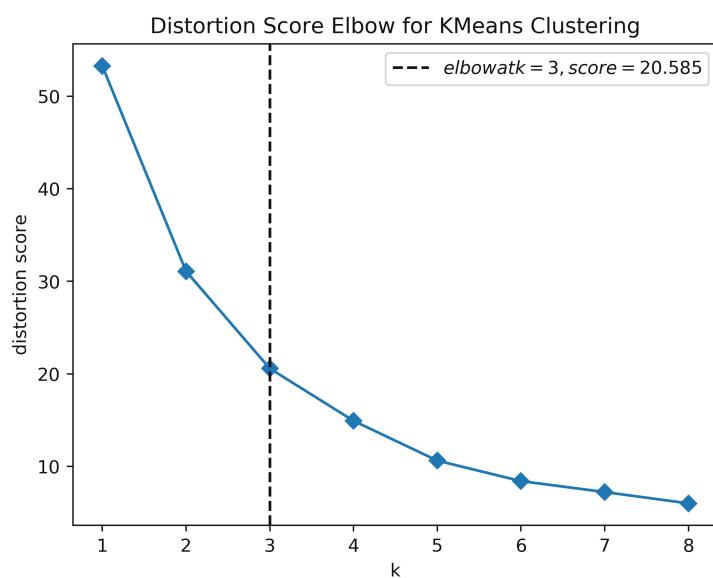


Figura 1.3: Resultado do método do cotovelo, com o gráfico de número de clusters x inércia dos centróides. O pacote yellowbrick.cluster foi utilizado para determinar o número de clusters ideal.

## Exercício 2 - Repita o exercício anterior considerando, entretanto, o algoritmo colorednoise.py

Os resultados das análises dos sinais referentes a este exercício se encontram na pasta **Exercise2**. O plot a seguir ilustra vinte sinais gerados para a família colornoise com  $\beta = 2$ , representando ruído marrom.

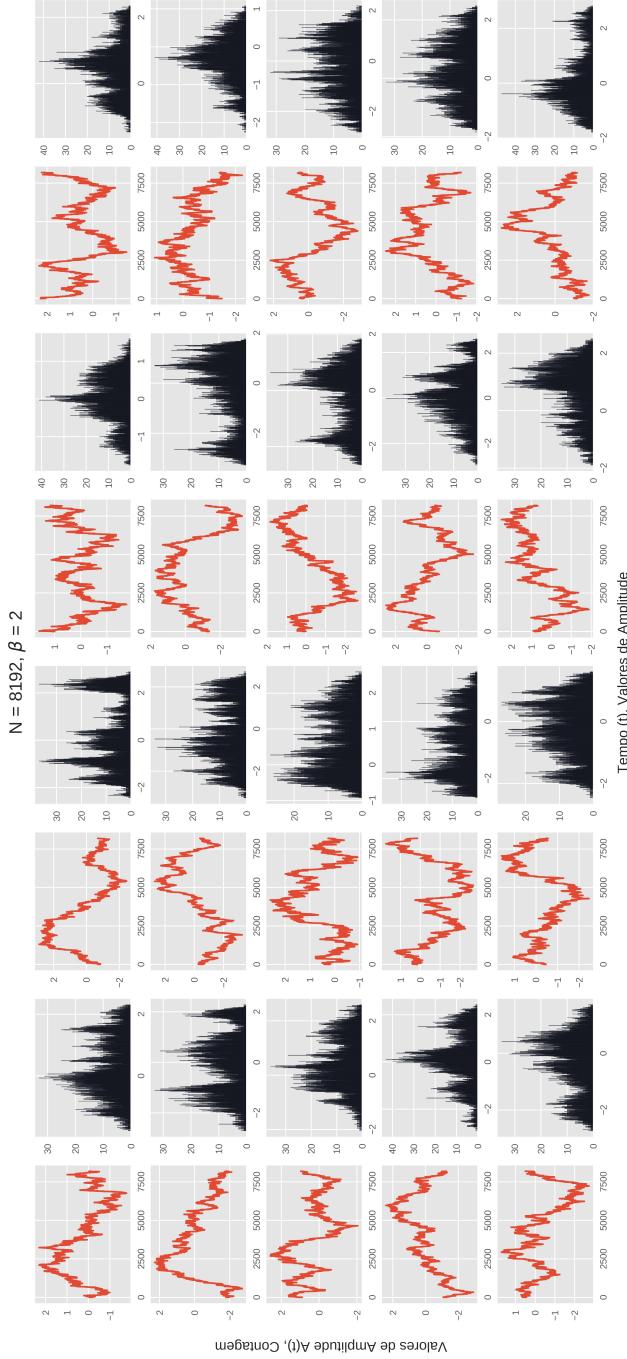


Figura 2.1: Série de 20 sinais diferentes gerados com o script colornoise.py. Ao todo neste exercício, três valores de  $\beta$  foram implementados: 0, gerando ruído branco, 1, do ruído rosa, e 2, representando ruído marrom, todas com tamanho da série igual a 8192. A figura exibe o resultado do ruído marrom e seus respectivos histogramas.

Os mesmos arquivos .csv gerados na primeira questão foram também gerados para a família colornois: *Exercicio2\_momentos.csv*, *Exercicio2\_parametros.csv* e *Exercicio2\_kmeans.csv*. Os resultados do agrupamento via kmeans são exibidos a seguir.

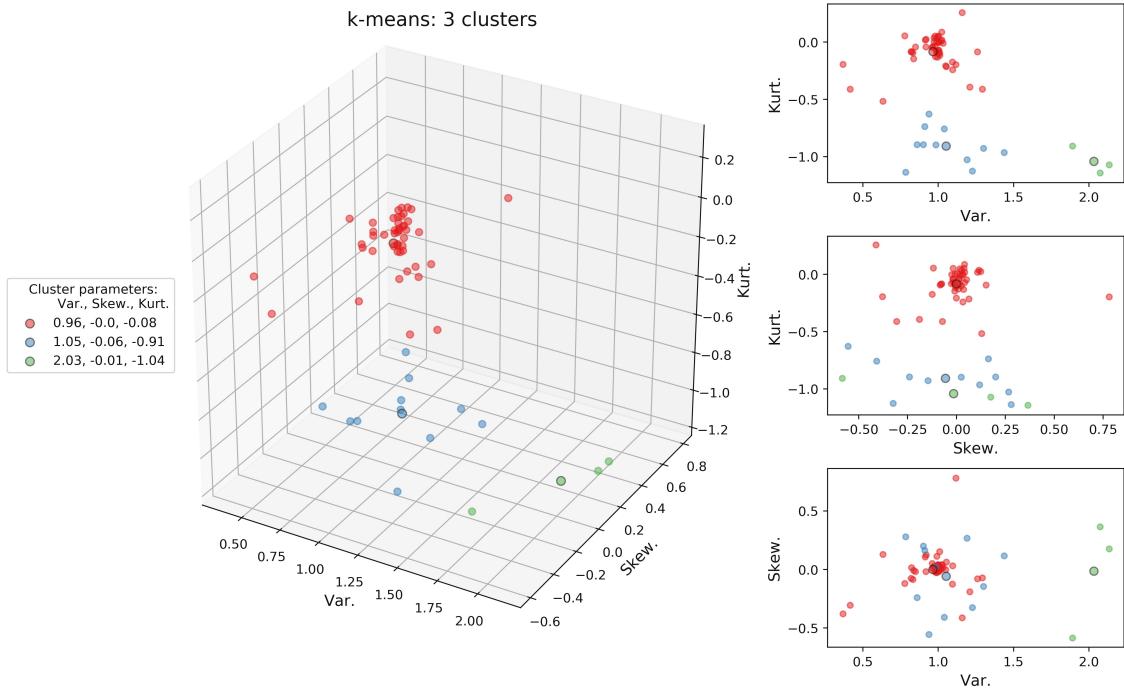


Figura 2.2: Técnica kmeans no espaço de parâmetros variância x skewness x kurtosis para a família de sinais colornoise. Resultado para número de clusters  $n\_c = 3$ .

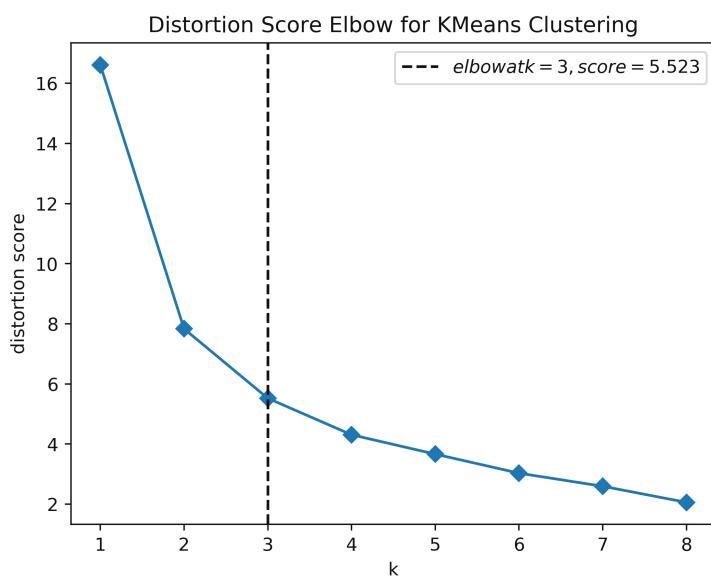


Figura 2.3: Resultado do método do cotovelo, mostrando que três clusters são necessários para agrupamento da família colornoise no espaço de parâmetros considerado.

### Exercício 3 - Repita o exercício anterior considerando, entre-tanto, o algoritmo pmodel.py

Os resultados das análises dos sinais referentes a este exercício se encontram na pasta **Exercise3**. Os valores de  $\rho$  da família pmnoise foram .18, .23, .28, .32, .37 e .42, onde os três primeiros representam uma série exógena (com parâmetro  $\beta = 0.7$ ), e os três últimos representam uma série endógena (com parâmetro  $\beta = 0.4$ ). Todas as séries foram geradas com tamanho igual a 8192. Os plots abaixo ilustram alguns dos resultados para a série exógena e endógena, respectivamente.

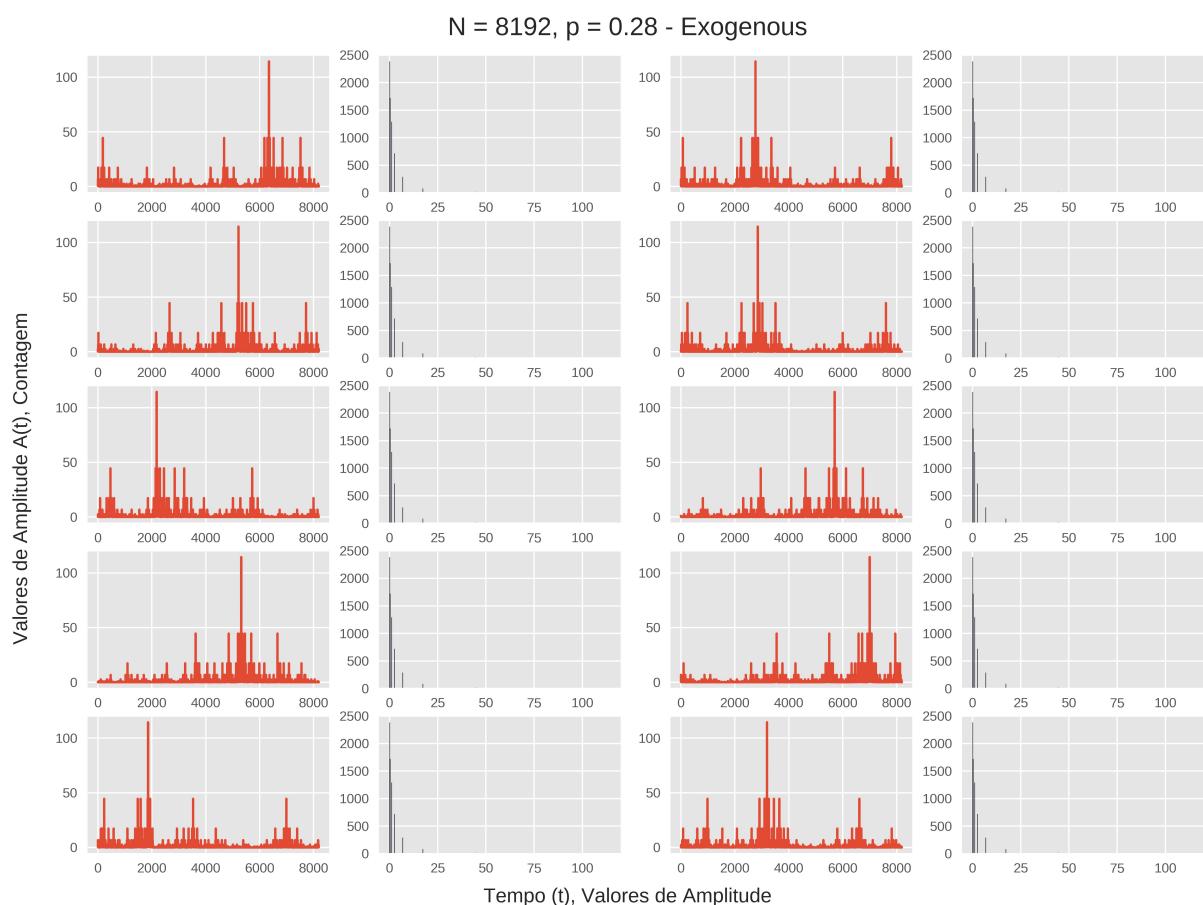


Figura 3.1: Plots de 10 sinais da séire exógena ( $\beta = 0.7$ ) da família pmnoise e seus respectivos histogramas.

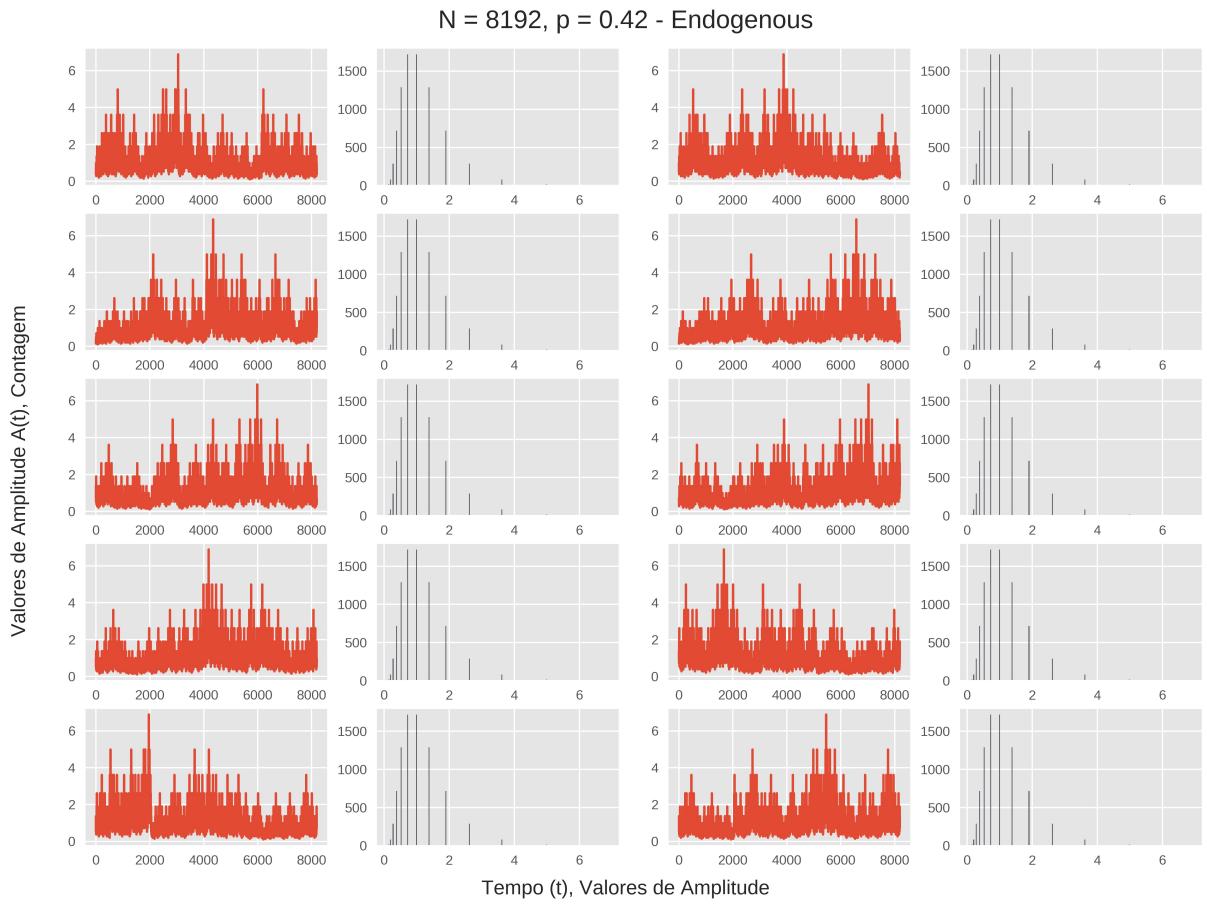


Figura 3.2: Plots de 10 sinais da séire endógena ( $\beta = 0.4$ ) da família pmnoise e seus respectivos histogramas.

Assim como para as famílias anteriores, a família pmnoise conta com os arquivos *Exercicio3\_momentos.csv*, *Exercicio3\_parametros.csv* e *Exercicio3\_kmeans.csv* em sua pasta, compondo o resultado da análise estatística em conjunto com os demais plots. Abaixo estão os resultados do agrupamento no espaço via kmeans. Fica evidente que há pouca variabilidade de cada sinal no espaço de parâmetros usado, de forma que o agrupamento não permite a caracterização de classes do grupo pmnoise no espaço composto por variância, skewness e kurtosis.

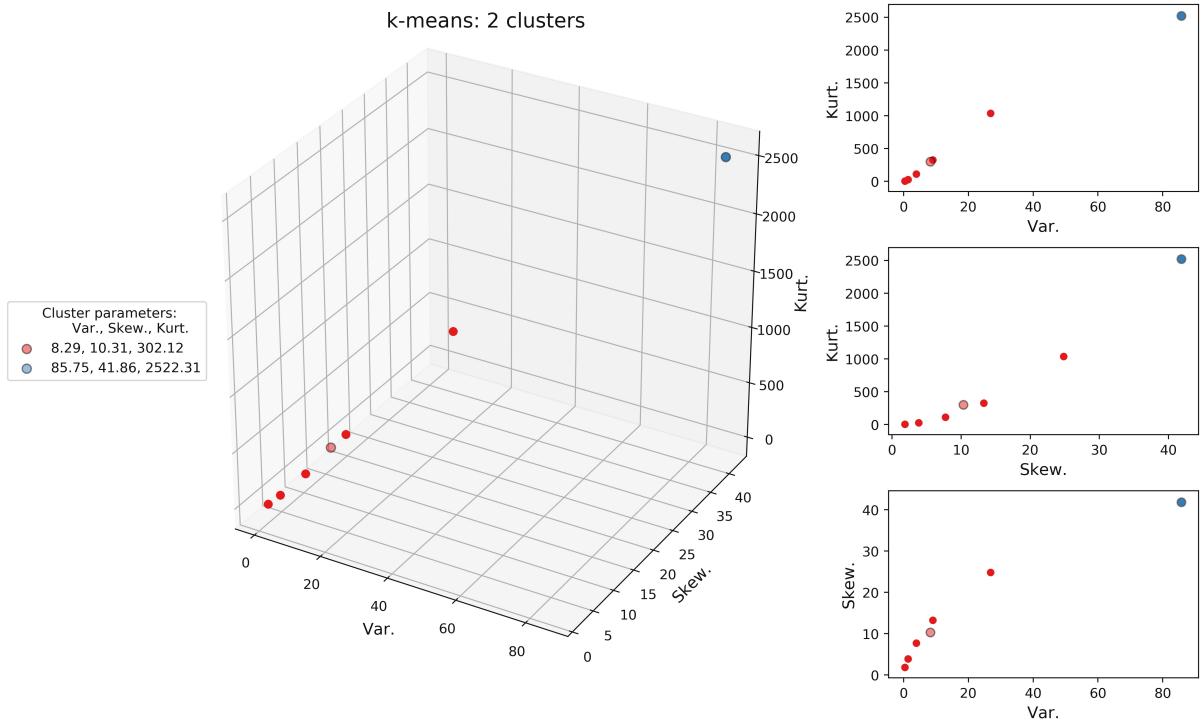


Figura 3.3: Técnica kmeans no espaço de parâmetros variância x skewness x kurtosis para todos os sinais da família de sinais pmnoise (endógenos e exógenos). Resultado para número de clusters  $n\_c = 2$ .

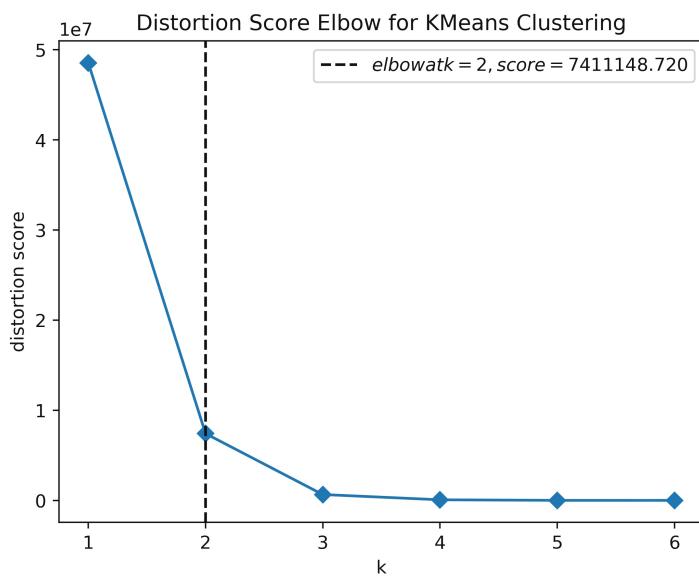


Figura 3.4: Resultado do método do cotovelo, mostrando que  $n\_c = 2$  obteve a melhor performance durante o agrupamento da família pmnoise no espaço de parâmetros considerado.

## Exercício 4 - Espaço de Cullen-Frey e Distribuições de Probabilidades

Os resultados deste exercício se encontram na pasta **Exercise4**. Na pasta **familia1** estão as análises do grupo noise para  $n = 8192$ , e na pasta **familia2** as análises do grupo colornoise para  $n = 8192$  e  $\beta = 0$ .

### 4.1

A análise desta Seção foi realizada com o script CullenFrey\_R\_Python.py.

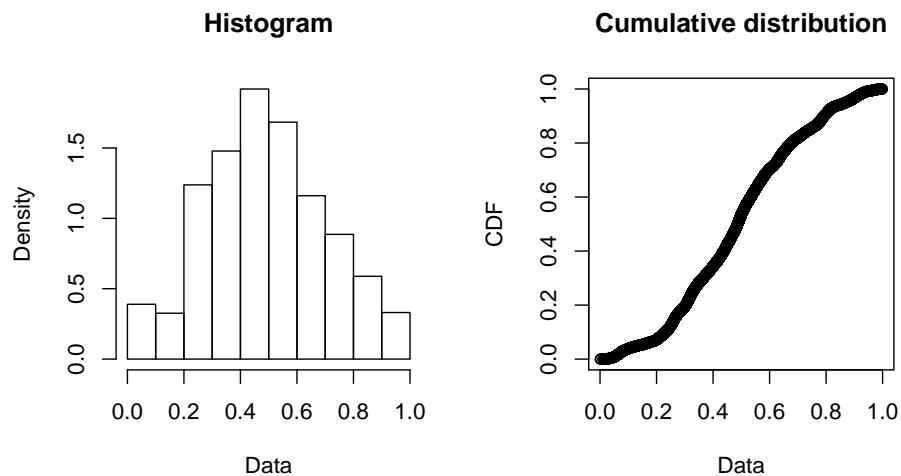


Figura 4.1.1: Histograma e CDF para a família 1 (noise) com  $n = 8192$ .

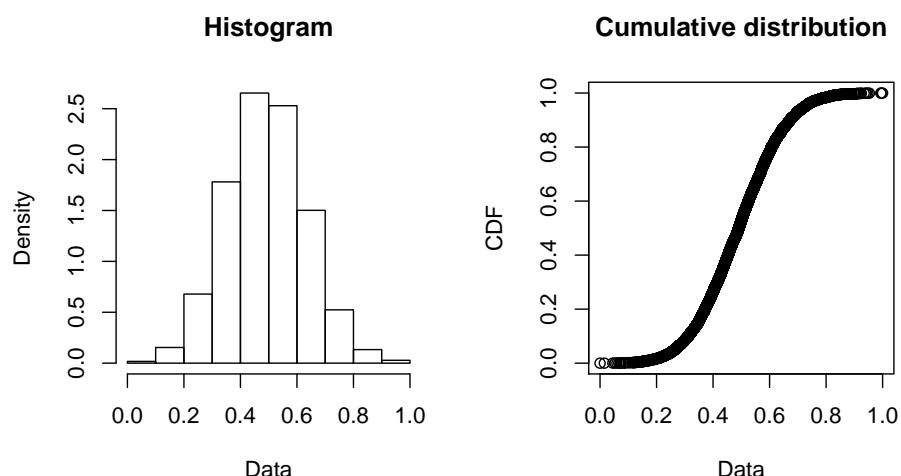


Figura 4.1.2: Histograma e CDF para a família 2 (colornoise) com  $n = 8192$  e  $\beta = 0$ .

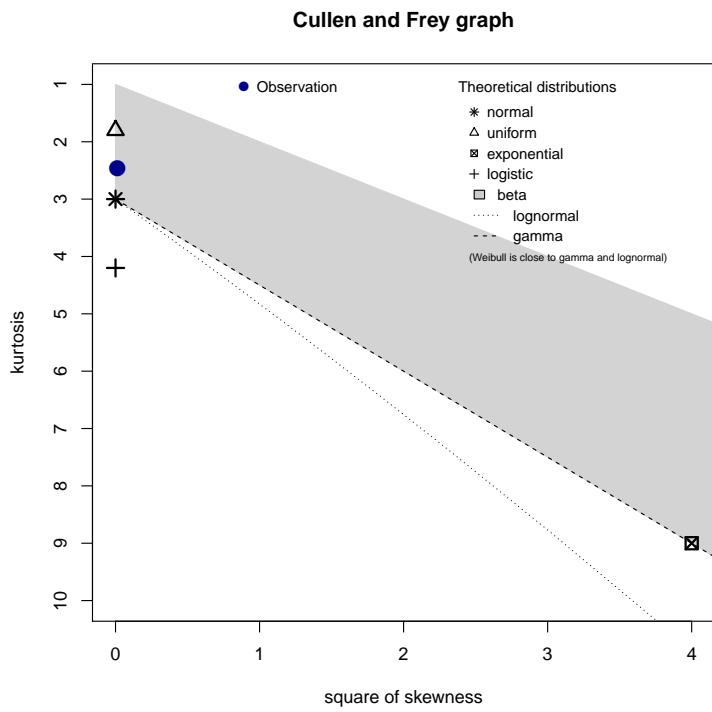


Figura 4.1.3: Resultado da análise Cullen and Frey para a família 1 (noise).

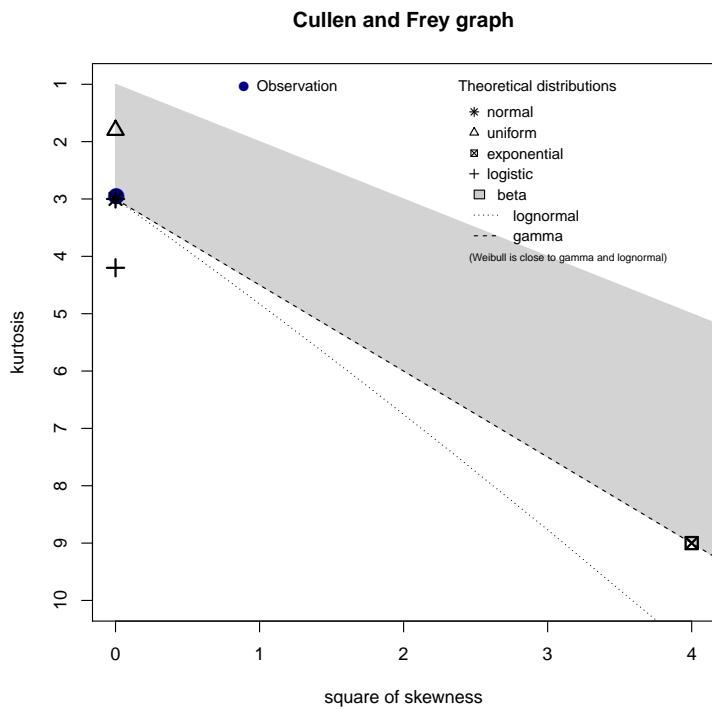


Figura 4.1.4: Resultado da análise Cullen and Frey para a família 2 (pmnoise).

## 4.2

Todos os resultados desta Seção forma gerados com os scripts Distribution\_Fitter\_Python.py e Distribution\_Fitter\_R.py.

### Primeira família

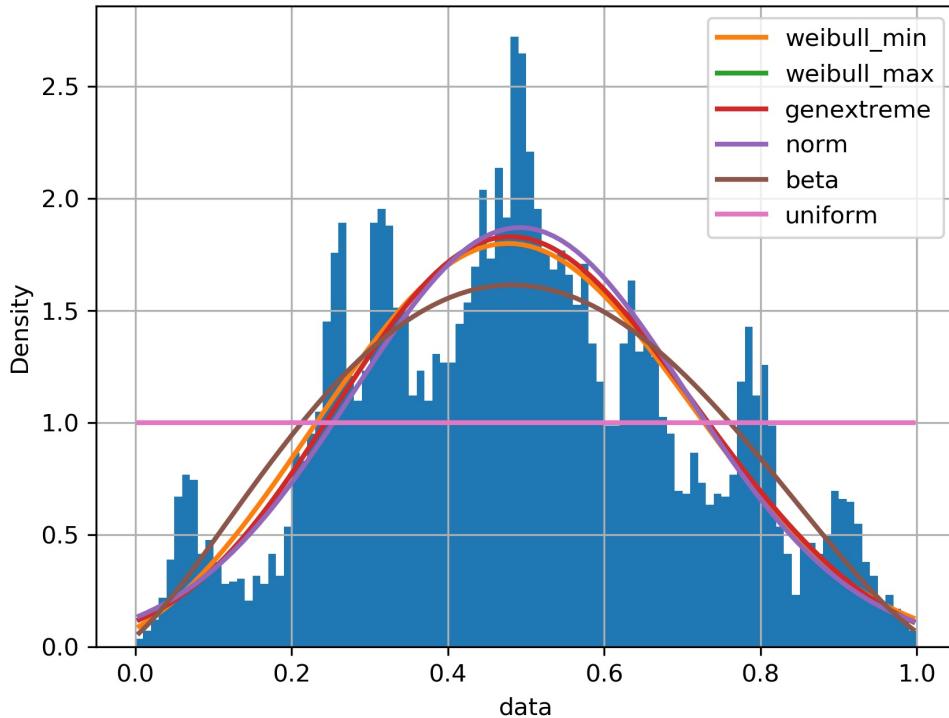


Figura 4.2.1: Resultado do ajuste de 6 distribuições diferentes, incluindo GEV, na família noise. Foi utilizado o script Distribution\_Fitter\_Python.py para este plot.

Abaixo o resultado do benchmark gerado automaticamente pelo script Distribution\_Fitter\_Python.py, presente no arquivo *Exercicio4\_2\_fam1\_Python.fits.txt*, que permite avaliar a performance de cada distribuição no ajuste empregado. Observa-se que a **weibull** foi a distribuição com menor erro.

Python FITs parameters:

	aic	bic	kl_div	sumsquare_error
weibull_min	60.026978	-54578.198214	0.055588	10.435733
weibull_max	61.235843	-54529.522935	0.054689	10.497924
genextreme	61.247047	-54529.316495	0.054691	10.498189
norm	62.123090	-54199.213029	0.056831	10.941890
beta	52.953609	-53710.669971	0.061723	11.588755
uniform	4.000000	-43825.167810	0.274875	38.821006

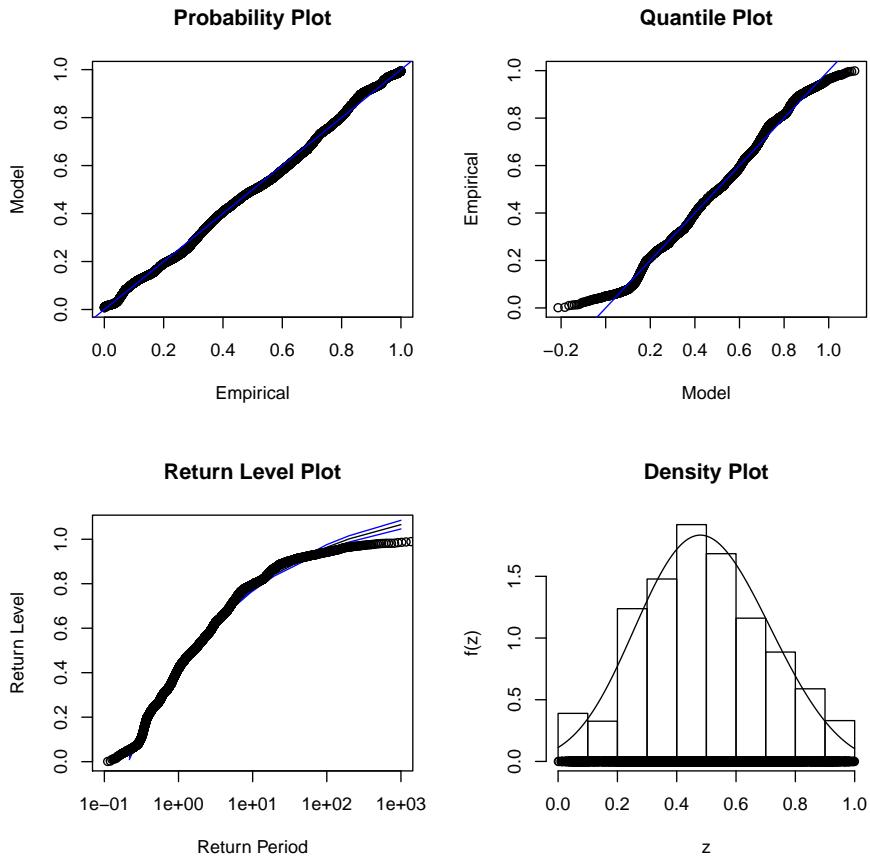


Figura 4.2.2: Resultado do ajuste da GEV com o script `Distribution_Fitter_R.py`. Um ajuste das mesmas distribuições da Figura 4.2.1 foi gerado, mas essa figura ilustra apenas a performance da GEV com o script com ambiente R em Python, que inclui plots dos quantis teóricos vs os empíricos, e das probabilidades teóricas vs as empíricas, além da curva da GEV sobre o histograma da série.

Abaixo o resultado do benchmark gerado automaticamente pelo script `Distribution_Fitter_R.py` para o ajuste de uma GEV, presente no arquivo `Exercicio4_2-fam1_GEV_params.txt`, com o resultado dos parâmetros da GEV via MLE (Maximum Likelihood Estimation) e seus respectivos erros.

GEV parameters:

Convergence code. Zero indicates successful convergence:  
0

Negative log-likelihood value:  
-1128.109

MLE's for the location, scale and shape parameters:  
0.4160513 0.2089967 -0.2728666

Standard errors for the MLE's for the location, scaleand shape parameters:  
0.002589172 0.001863585 0.008340027

## Segunda família

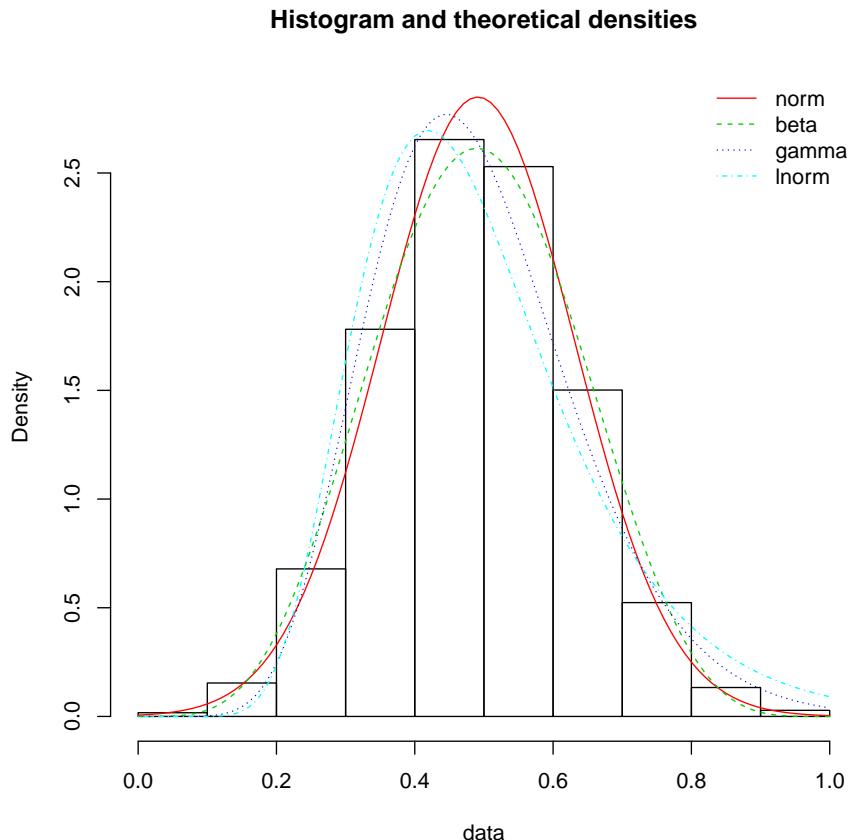


Figura 4.2.3: Resultado do ajuste de 4 distribuições diferentes na família pmnoise. Foi utilizado o script `Distribution_Fitter_R.py` para este plot. Não foi ajustada uma GEV para esta família.

Abaixo o resultado do benchmark gerado automaticamente pelo script `Distribution_Fitter_R.py`, presente no arquivo `Exercicio4_2_fam1_gof.txt`, com o resultado dos ajustes através do método `mle` (maximum likelihood estimation), que é o padrão do código. Observa-se que a distribuição `normal` foi a de melhor performance.

```
Goodness-of-fit statistics
                           1-mle-norm 2-mle-beta 3-mle-gamma 4-mle-lnorm
Kolmogorov-Smirnov statistic 0.008831258 0.02152509 0.04381156 0.06334796
Cramer-von Mises statistic   0.114916948 0.97570799 5.00102538 13.55002675
Anderson-Darling statistic   0.721170509          Inf 30.73308136 84.89555447

Goodness-of-fit criteria
                           1-mle-norm 2-mle-beta 3-mle-gamma 4-mle-lnorm
Akaike's Information Criterion -8960.078 -8705.655 -8355.618 -7135.150
Bayesian Information Criterion -8946.057 -8691.634 -8341.596 -7121.129
```

## Exercício 5 - Processos Estocásticos Canônicos: Caos Determinístico e Turbulência

Os resultados deste exercício se encontram na pasta **Exercise4**, junto com todas as análises da família chaosnoise, separadas entre a pasta **Logístico** e **Hénon**.

### 5.1

#### Logístico

Para o mapeamento Logístico, os valores de  $\rho$  escolhidos foram 3.81, 3.905 e 4.

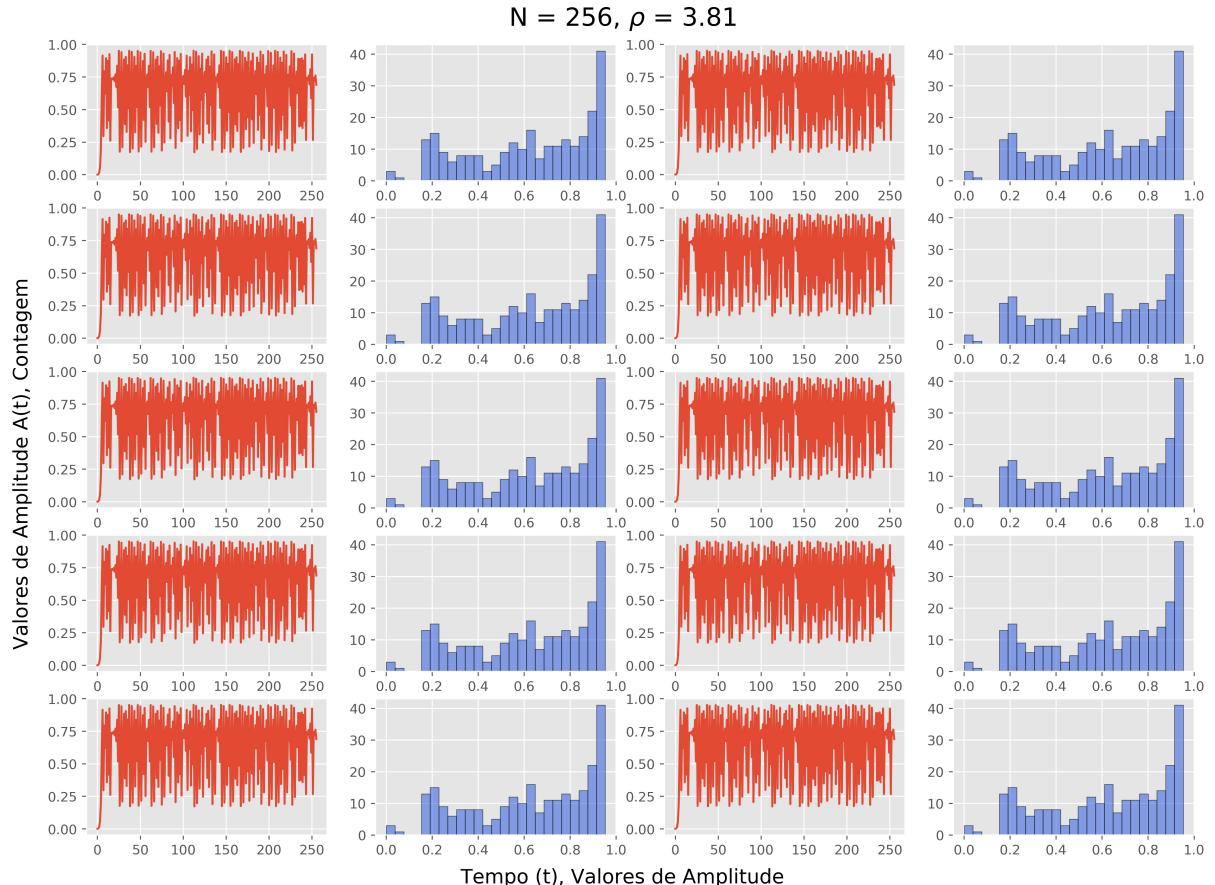


Figura 5.1.1: Plots de 10 sinais para o mapeamento Logístico da família chaosnoise e seus respectivos histogramas. O tamanho da série foi de 256 para todos os valores de  $\rho$ . Acima os resultados para  $\rho = 3.81$ .

Os mesmos scripts da quarta questão foram implementados nas análises da família chaosnoise. A seguir são apresentados a classificação no espaço de Cullen and Frey e um ajuste de distribuições (incluindo GEV) a um dos sinais do mapeamento Logístico com  $\rho = 3.81$  (mais precisamente, o último sinal da Figura 5.1.1).

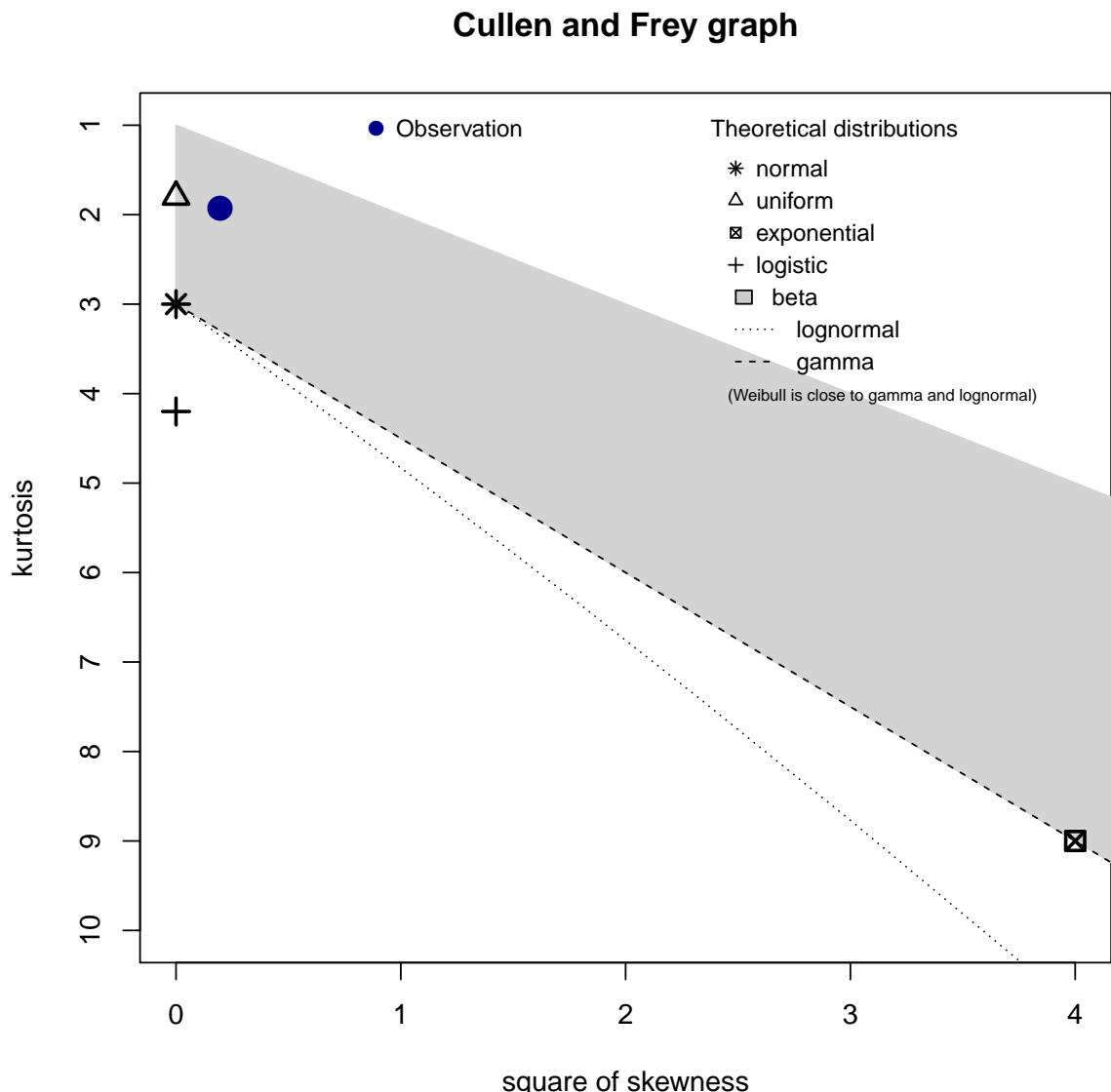


Figura 5.1.2: Resultado da análise no espaço de Cullen and Frey para o mapeamento Logístico com  $N = 256$  e  $\rho = 3.81$  (último sinal da Figura 5.1.1).

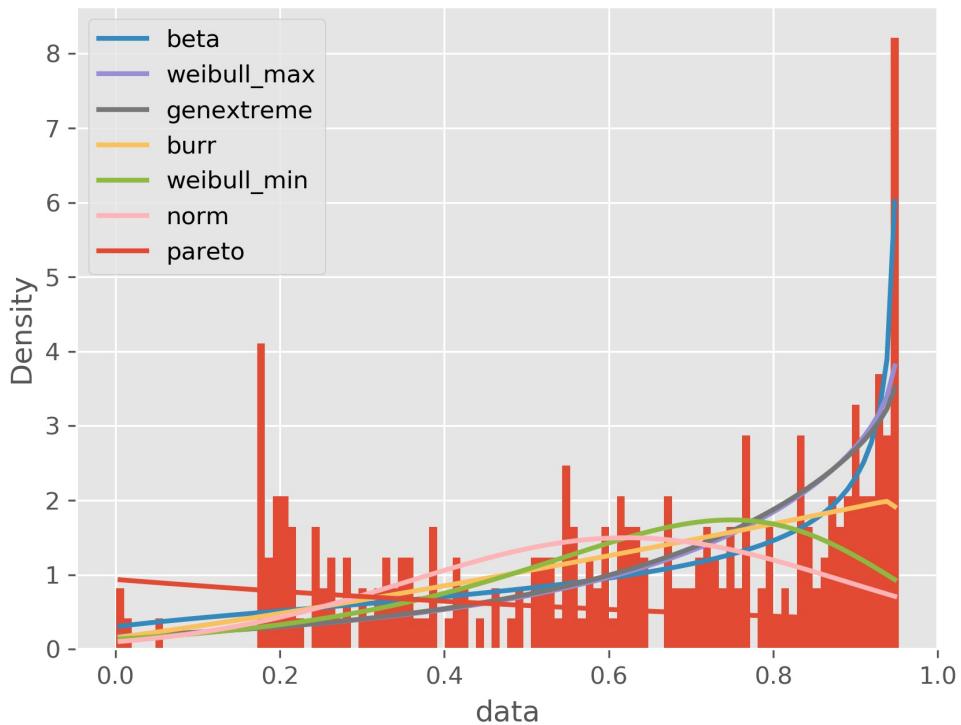


Figura 5.1.3: Resultado do ajuste de 7 distribuições ao último sinal presente na Figura 5.1.1.

Resultado do benchmark do ajuste das distribuições. Observa-se que a distribuição **beta** foi a de melhor ajuste. Esse resultado se encontra no arquivo *rho\_3.81-Python.fits.txt*.

**Python FITs parameters:**

	aic	bic	k1_div	sumsquare_error
beta	43.442952	-347.070264	inf	60.508922
weibull_max	78.687389	-265.208574	inf	85.133780
genextreme	76.853924	-256.519561	inf	88.072942
burr	39.120899	-215.028318	inf	101.349980
weibull_min	67.478982	-154.804792	inf	131.038173
norm	59.167005	-144.201196	inf	139.570496
pareto	108.265130	-88.650689	inf	169.678105

## Henon

Para o mapeamento de Henon, os valores de  $a$  escolhidos foram 1.32 e 1.4. Para  $b$ , escolheu-se os valores 0.21, 0.26 e 0.31, de forma a fixar o primeiro e variar o segundo gerando-se cinco sinais para cada combinação de parâmetros.

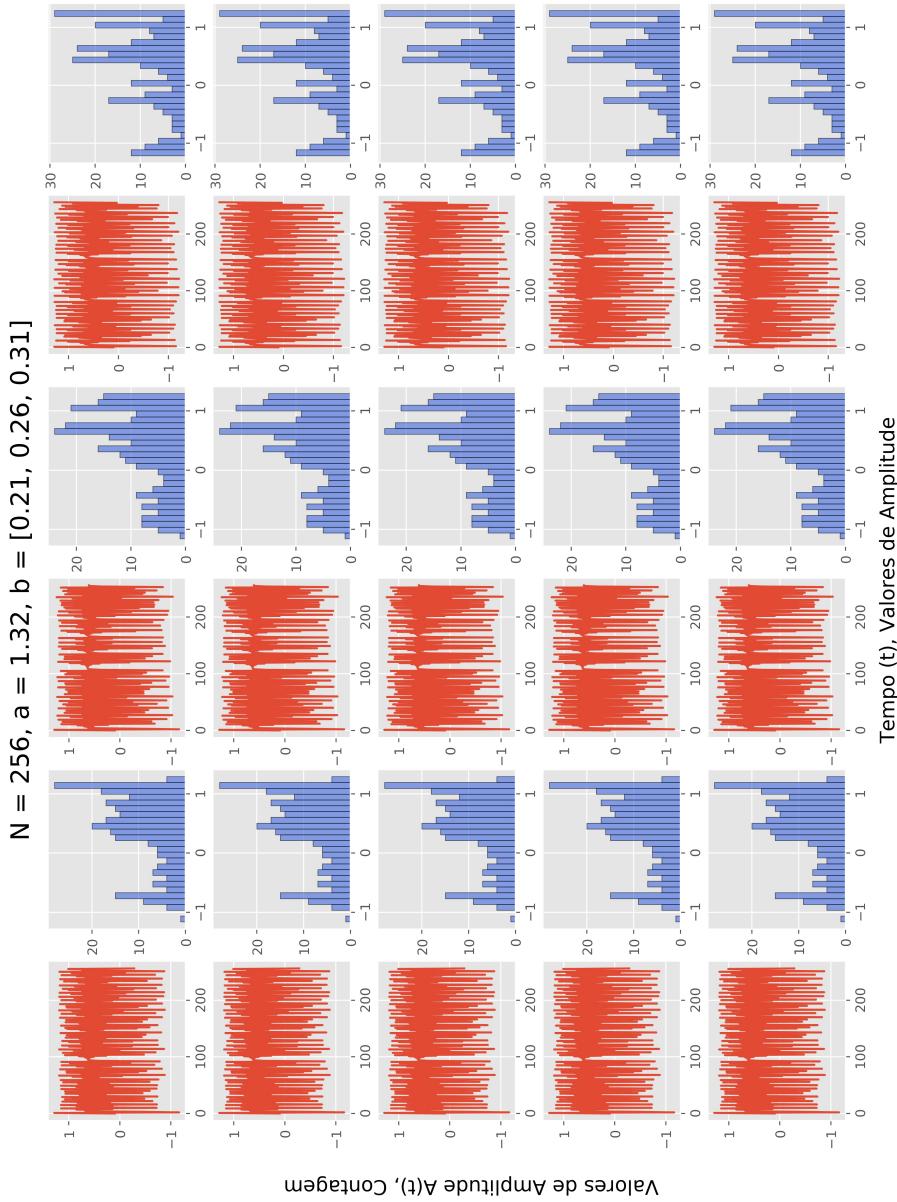


Figura 5.1.4: Série de 15 sinais diferentes (e seus respectivos histogramas) gerados com o script `chaosnoise.py`. A figura exibe o resultado de  $a = 1.32$  e três valores de  $b$ . Os cinco primeiros sinais (da esquerda) são para  $b = 0.21$ , os cinco do centro para  $b = 0.26$ , e os cinco da direita para  $b = 0.31$ .

A seguir são apresentados a classificação no espaço de Cullen and Frey e um ajuste de distribuições (incluindo GEV) a um dos sinais do mapeamento de Henon com  $a = 1.32$  e  $b = 0.31$  (mais precisamente, o último sinal da Figura 5.1.4).

### Cullen and Frey graph

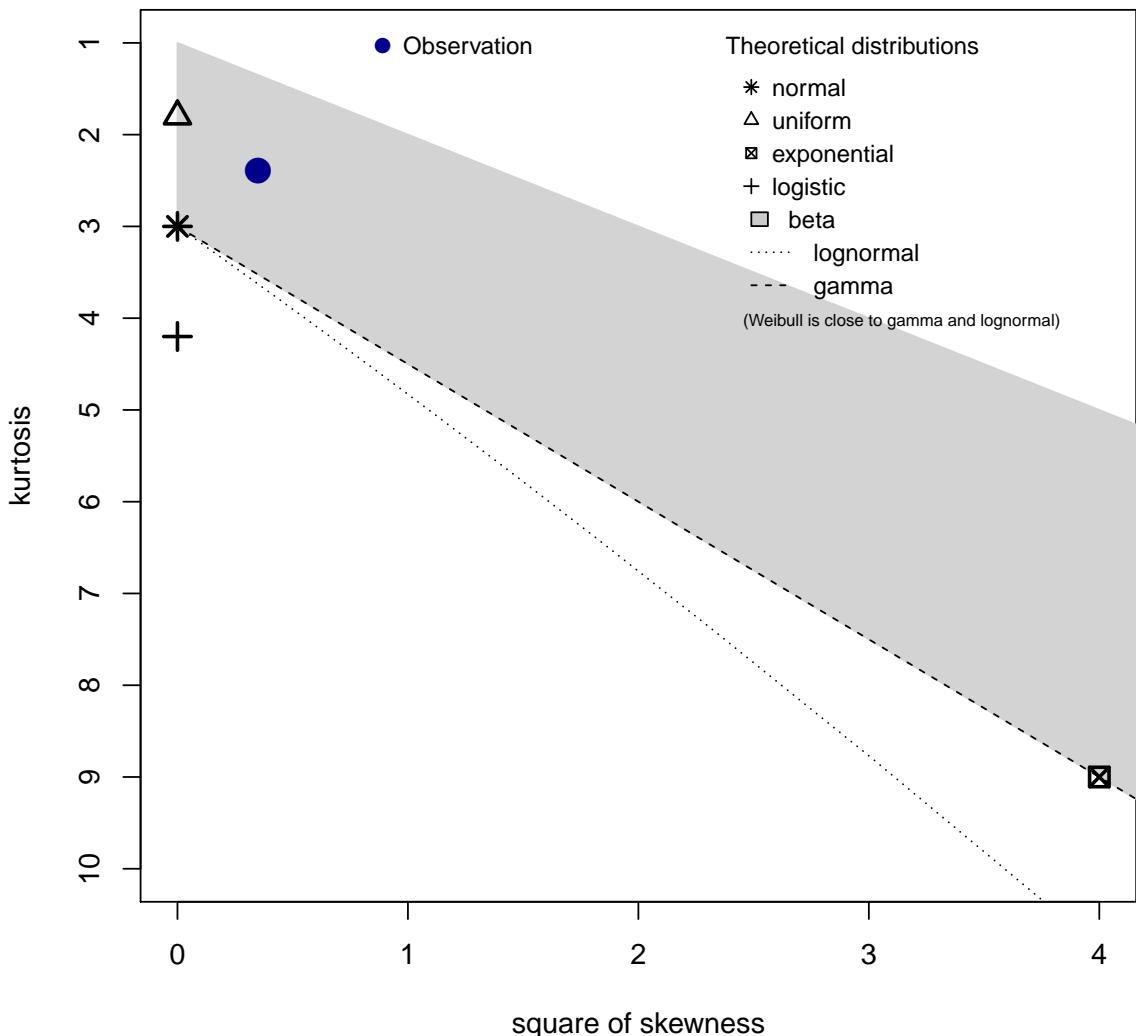


Figura 5.1.5: Resultado da análise no espaço de Cullen and Frey para o mapeamento de Hénon com  $a = 1.32$  e  $b = 0.31$  (último sinal da Figura 5.1.4).

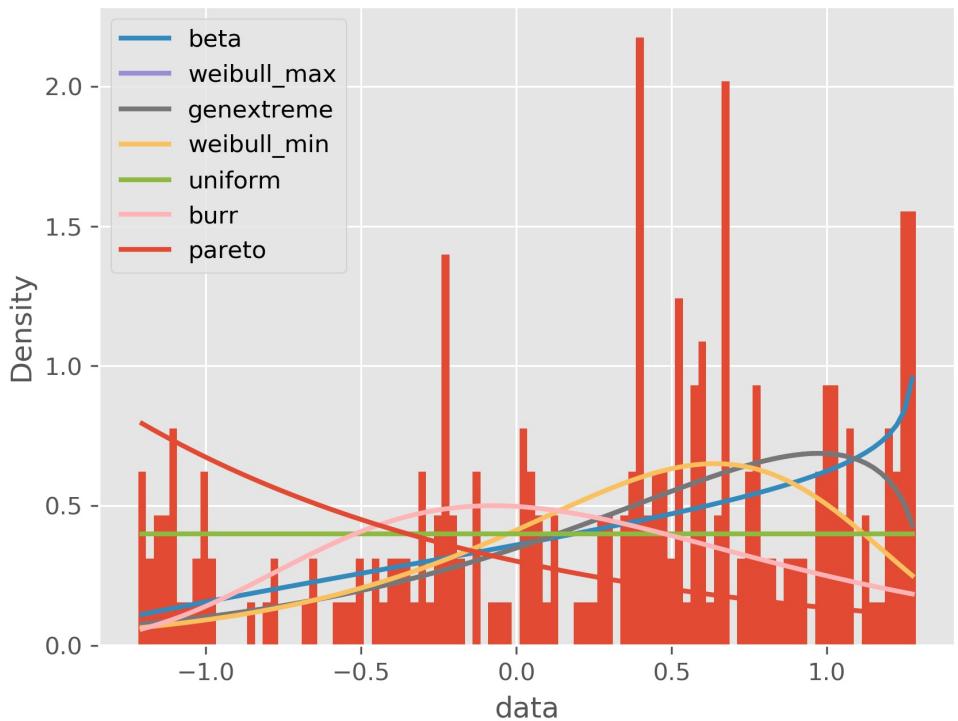


Figura 5.1.6: Resultado do ajuste de 7 distribuições ao último sinal presente na Figura 5.1.4.

Resultado do benchmark do ajuste das distribuições. Novamente, observa-se que a distribuição `beta` foi a de melhor ajuste. Esse resultado se encontra no arquivo `a_1.32_b_0.31_Python.fits.txt`.

`Python FITs parameters:`

	aic	bic	kl_div	sumsquare_error
<code>beta</code>	220.628587	-708.130851	inf	14.989294
<code>weibull_max</code>	240.785844	-689.835893	inf	16.446530
<code>genextreme</code>	240.782424	-689.835360	inf	16.446564
<code>weibull_min</code>	246.146689	-681.695092	inf	16.975834
<code>uniform</code>	187.690059	-677.673554	inf	17.619928
<code>burr</code>	245.196775	-631.880639	inf	20.166637
<code>pareto</code>	251.063461	-564.684343	inf	26.764745

## 5.2

A segunda lei de Newton é definida por:

$$\sum \vec{f}_{ext} = m\vec{a},$$

que representa que a aceleração de um corpo é devida ao somatório das forças externas aplicadas à ele. Para uma parcela de fluido de densidade  $\rho$ , a força por unidade de volume é

$$\sum \vec{f} = \rho\vec{a} = \vec{f}_{grav} + \vec{f}_{press} + \vec{f}_{visc},$$

onde  $\vec{f}_{grav} = \rho\vec{g}$  é a força gravitacional, agindo sobre o fluido na direção negativa de  $\vec{z}$ , ou seja, para baixo;  $\vec{f}_{press} = -\vec{\nabla}p$  representa as forças de pressão, agindo sobre o fluido em todas as direções, para o seu interior e normal à sua superfície;  $\vec{f}_{visc} = \mu\nabla^2\vec{v}$  denota a força viscosa, que é proporcional à velocidade  $v$  do fluido e age em sua superfície (normalmente ou tangencialmente) por conta da viscosidade  $\mu$  do mesmo ( $\nabla^2$  é o operador Laplaciano).

Expandindo a aceleração, escrevendo-a em termos de derivadas dos componentes da velocidade, e substituindo cada um dos componentes de força atuando sobre a parcela de fluido, temos

$$\rho \left( \frac{\partial \vec{v}}{\partial t} + \vec{v}_x \frac{\partial \vec{v}}{\partial x} + \vec{v}_y \frac{\partial \vec{v}}{\partial y} + \vec{v}_z \frac{\partial \vec{v}}{\partial z} \right) = \rho\vec{g} - \vec{\nabla}p + \mu\nabla^2\vec{v},$$

que é a equação de *Navier-Stokes* válida para fluidos Newtonianos incompressíveis.

## Exercício 6 - PSD & DFA

Os resultados deste exercício se encontram na pasta **Exercise6** organizados nas pastas **6.1**, **6.2** e **6.3**.

### 6.1.

Como exemplo, a seguir estão os resultados da análise do grupo noise, presentes na pasta **grupo\_noise** (dentro da pasta referente a este exercício - **6.1**).

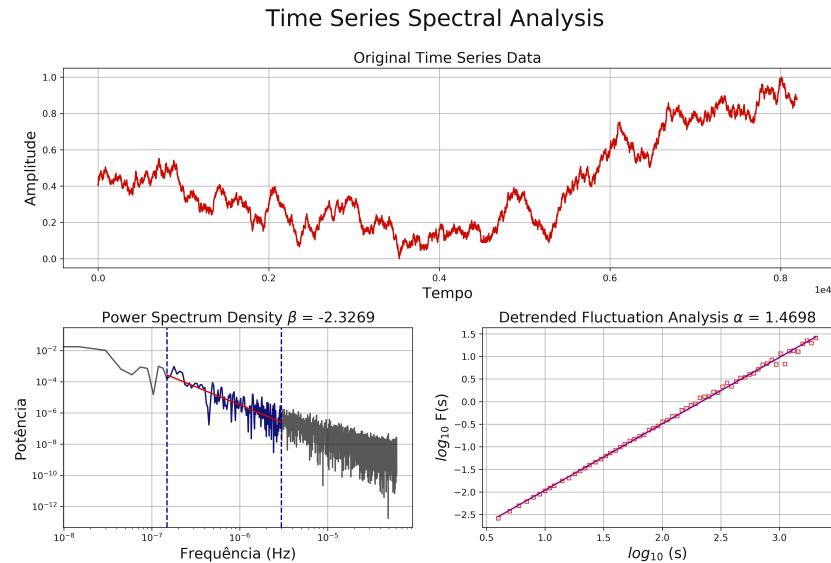


Figura 6.1.1: Acima, sinal do grupo noise com  $n = 8192$ ; abaixo à esquerda, o PSD e o índice espectral  $\beta \sim -2.32$ ; abaixo à direita, o DFA com o expoente  $\alpha \sim 1.47$ .

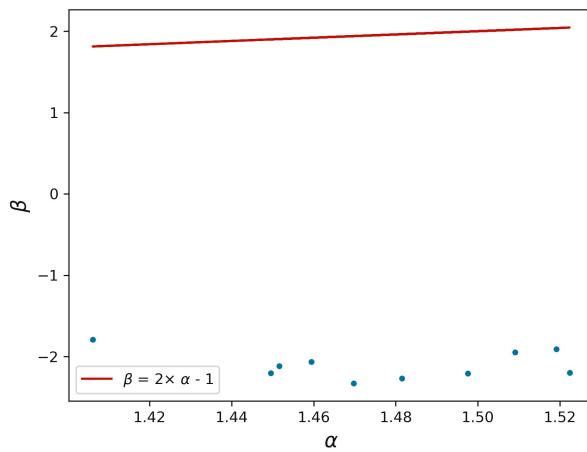


Figura 6.1.2: Resultado do teste do ajuste dos valores de  $\beta$  e  $\alpha$  para a família noise com  $n = 8192$  à equação do Teorema de Wiener–Khinchin-Parseval. Os pontos são valores empíricos. A reta é o resultado da equação do Teorema de WKP sobre os  $\alpha$ 's empíricos. Observa-se que, em módulo, o  $\beta$  teórico é compatível com o empírico.

Primeiro resultado do agrupamento kmeans do grupo noise (usando  $\beta$ ):

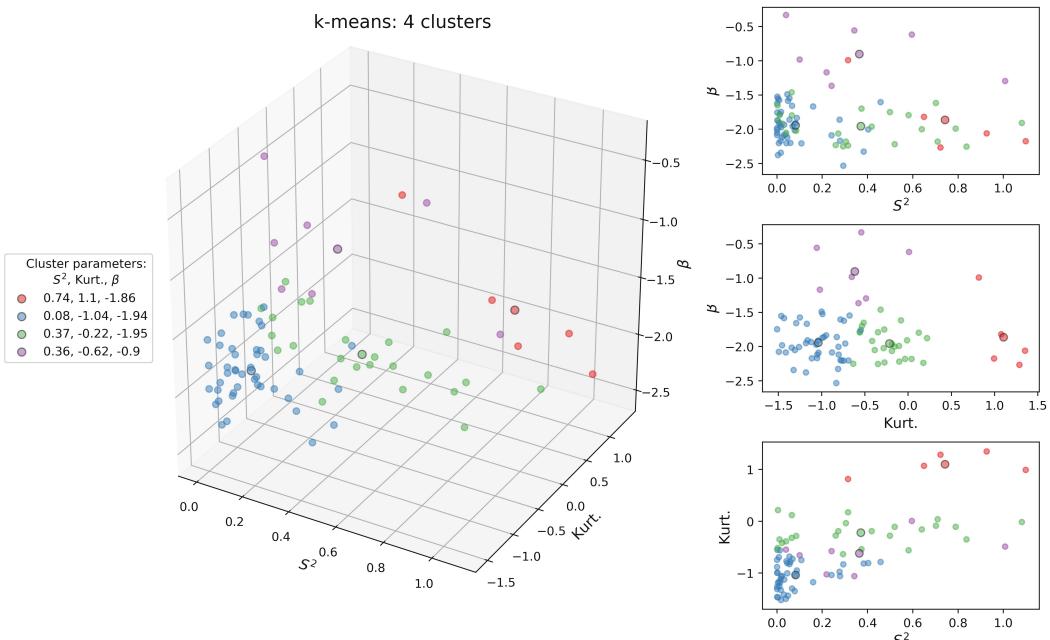


Figura 6.1.3: Técnica kmeans no espaço de parâmetros skewness<sup>2</sup> x kurtosis x  $\beta$  para toda a família noise, cujo melhor resultado foi  $n\_c = 4$  (quatro clusters).

Segundo resultado do agrupamento kmeans do grupo noise (usando  $\alpha$ ):

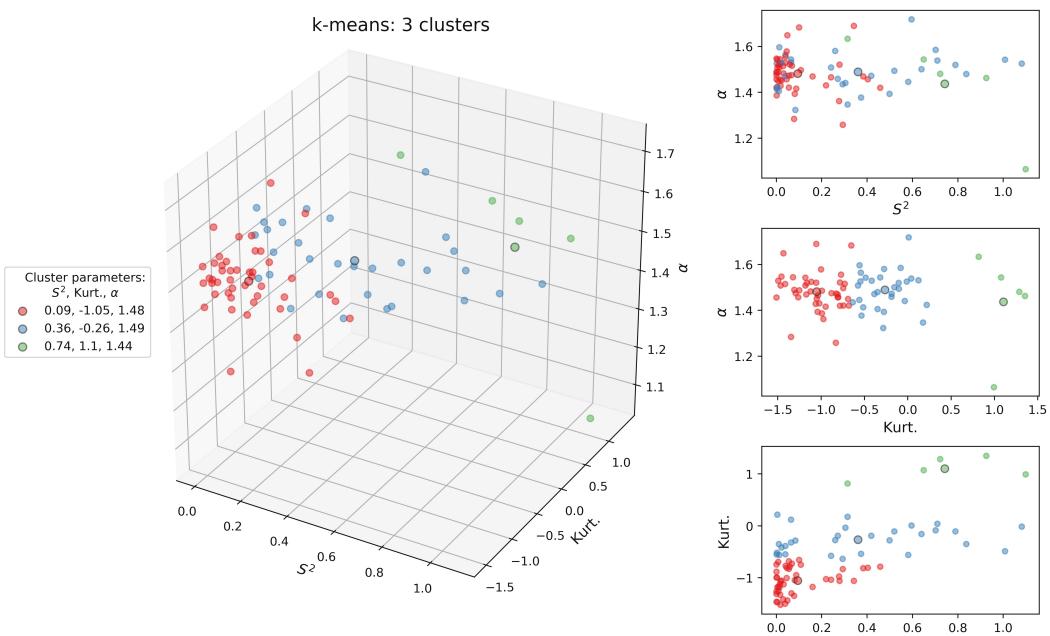


Figura 6.1.4: Técnica kmeans no espaço de parâmetros skewness<sup>2</sup> x kurtosis x  $\alpha$  para toda a família noise, cujo melhor resultado foi  $n\_c = 3$  (três clusters).

Importante ressaltar que as análises espetrais do grupo colornoise foram consistentes com as cores de cada ruído ( $\beta = 0, 1$  e  $2$ ). A seguir está um plot da análise para  $\beta = 1$  (Figura 6.1.5). Por outro lado, a análise espectral do grupo chaosnoise apresentou índice espectral com faixa dinâmica extremamente "lisa", similar a um ruído branco. Isso ocorre pois as séries do grupo chaosnoise, como o mapeamento Logístico (Figura 6.1.6), pertencem a regimes antipersistentes.

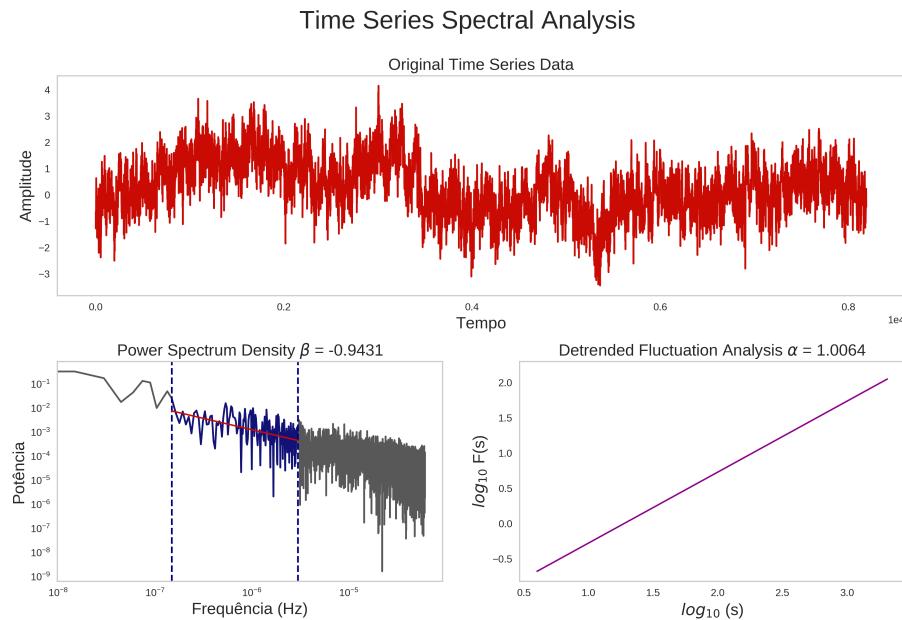


Figura 6.1.5: Análise espectral do grupo colornoise para  $\beta = 1$ .

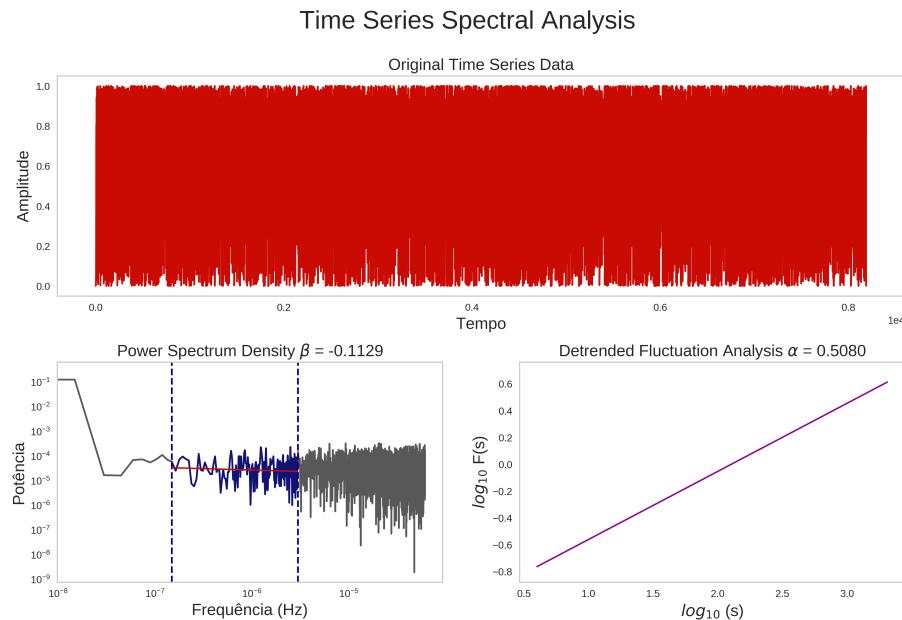


Figura 6.1.6: Análise espectral do grupo chaosnoise para o mapeamento Logístico com  $\rho = 4$ .

## 6.2

Para este exercício foi utilizado o script `kmeans_3D_plus_data.py`. Os resultados de cada família estão em suas respectivas pastas dentro da pasta referente a este exercício - **6.2**. As séries temporais *ST-Sol3GHz* e *ST-surftemp504* foram importadas da pasta `time_series_data`. A *ST-OWS\_NDC\_Covid19* é referente ao Brasil e foi baixada na execução do código.

Os plots a seguir foram escolhidos por exibir os resultados mais interessantes. Cada família está identificada com a cor vermelha (representando um cluster único, ou seja,  $n_c = 1$ ) e as séries temporais acima citadas são plotadas no mesmo espaço de parâmetros. Com isso pode-se identificar como as séries temporais se posicionam nos espaços considerados com relação aos sinais presentes na tabela *Dataset\_signal*. De todas as famílias, apenas *chaosnoise* não permitiu identificar alguma das três séries temporais como pertencente ao cluster da família. Para o grupo *noise* (Figura 6.2.1), a série *ST-surftemp504* esteve próxima dos sinais da família no espaço  $\text{skewness}^2 \times \text{kurtosis} \times \beta$ , mas nenhuma série temporal exibiu o mesmo comportamento no espaço  $\text{skewness}^2 \times \text{kurtosis} \times \alpha$ . Para os grupos *colornoise* (Figura 6.2.2) e *pmnoise* (Figura 6.2.3), o comportamento das séries temporais com relação aos sinais foi basicamente o mesmo para ambos os espaços de parâmetros considerados.

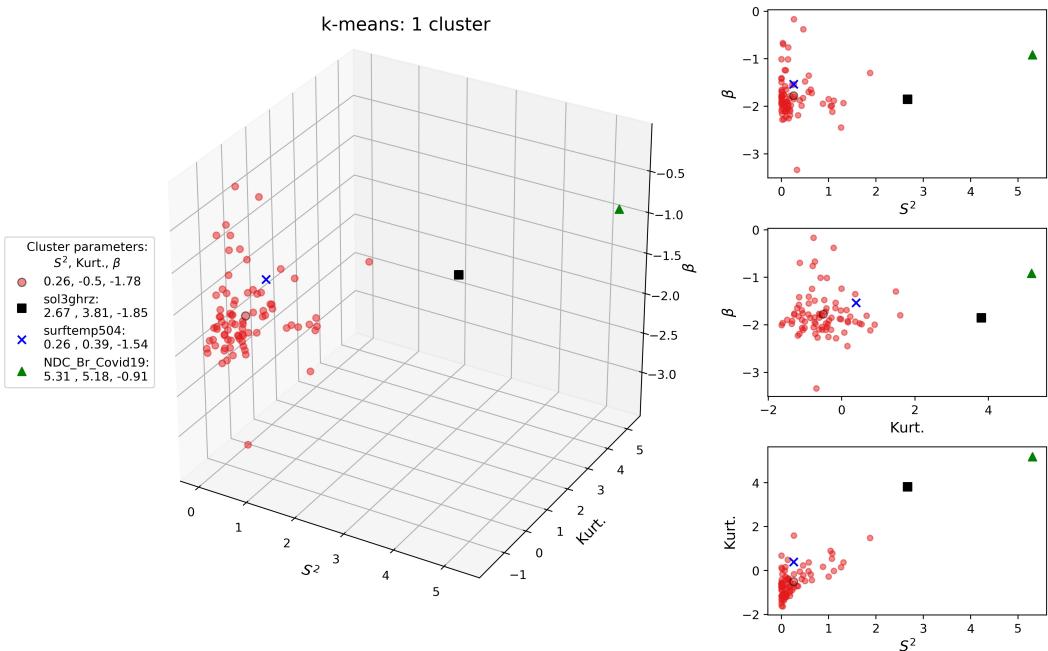


Figura 6.2.1: Espaço  $\text{skewness}^2 \times \text{kurtosis} \times \beta$  do grupo *noise* e a posição relativa das três séries temporais deste exercício. A série *ST-surftemp504* foi a que mais se aproximou dos sinais deste grupo.

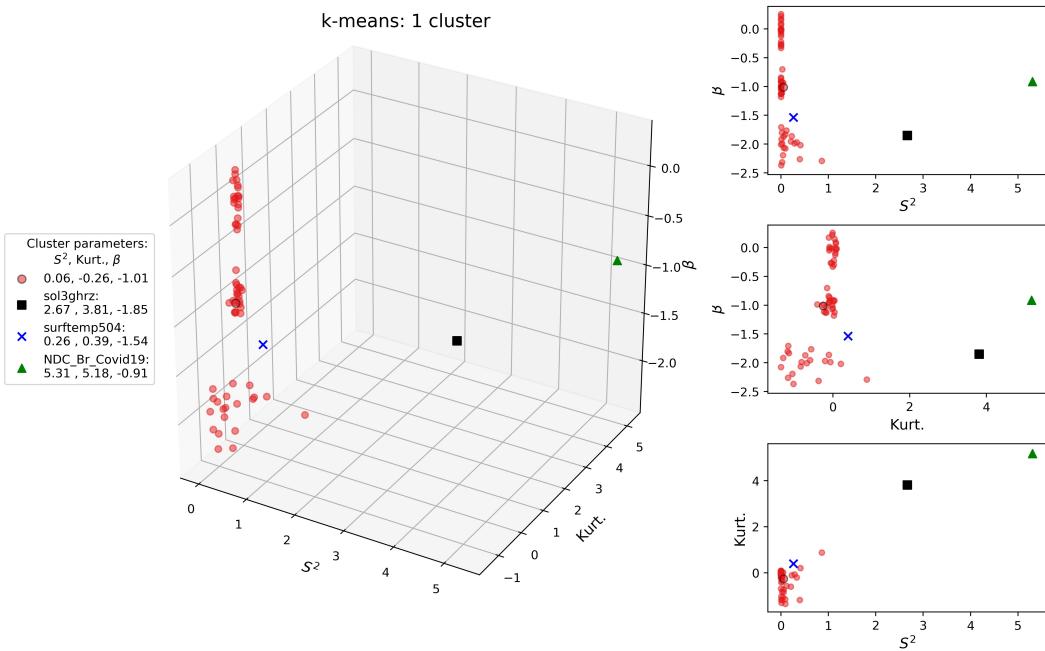


Figura 6.2.2: Espaço skewness<sup>2</sup> x kurtosis x  $\beta$  do grupo colornoise e a posição relativa das três séries temporais deste exercício. A série ST-surftemp504 foi a que mais se aproximou dos sinais deste grupo.

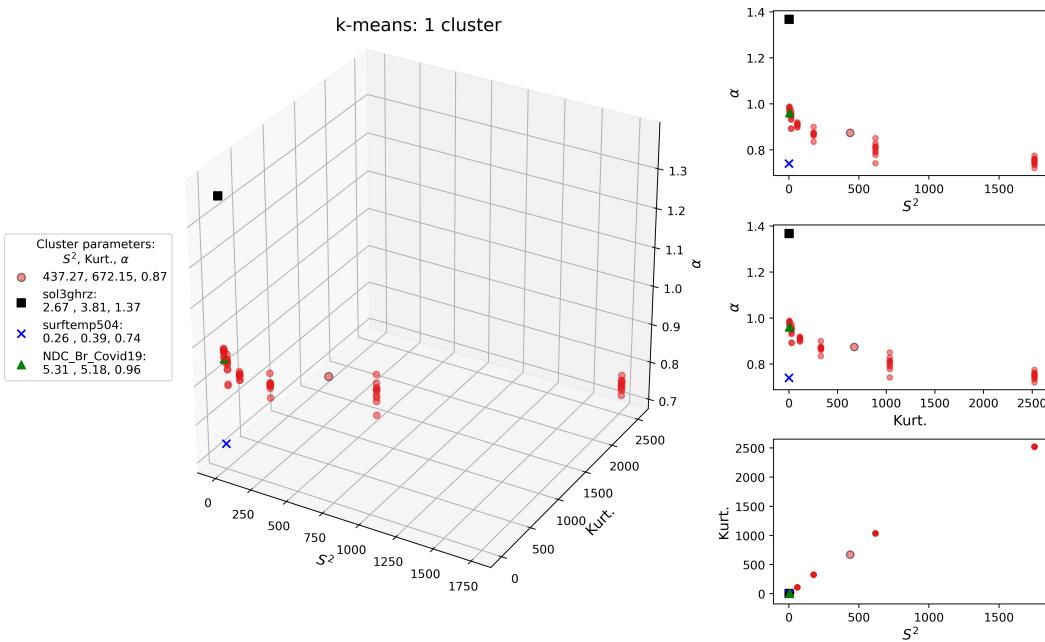


Figura 6.2.3: Espaço skewness<sup>2</sup> x kurtosis x  $\alpha$  do grupo pmnoise e a posição relativa das três séries temporais deste exercício. A série NDC\_Br\_Covid19 foi a que mais se aproximou dos sinais deste grupo.

### 6.3

Os dois plots a seguir resultam da análise da série *ST-OWS\_NDC\_Covid19* para todos os países. O número de clusters ideal foi selecionado de acordo com o método do cotovelo, e foi de três para os dois espaços de parâmetros deste exercício. O algoritmo utilizado foi o `kmeans_2D_group_flags.py`, que agrupou cada país em seu respectivo cluster em arquivos .csv (identificados pelo centróide do grupo). Estes arquivos e demais plots se encontram na pasta deste exercício - **6.3**.

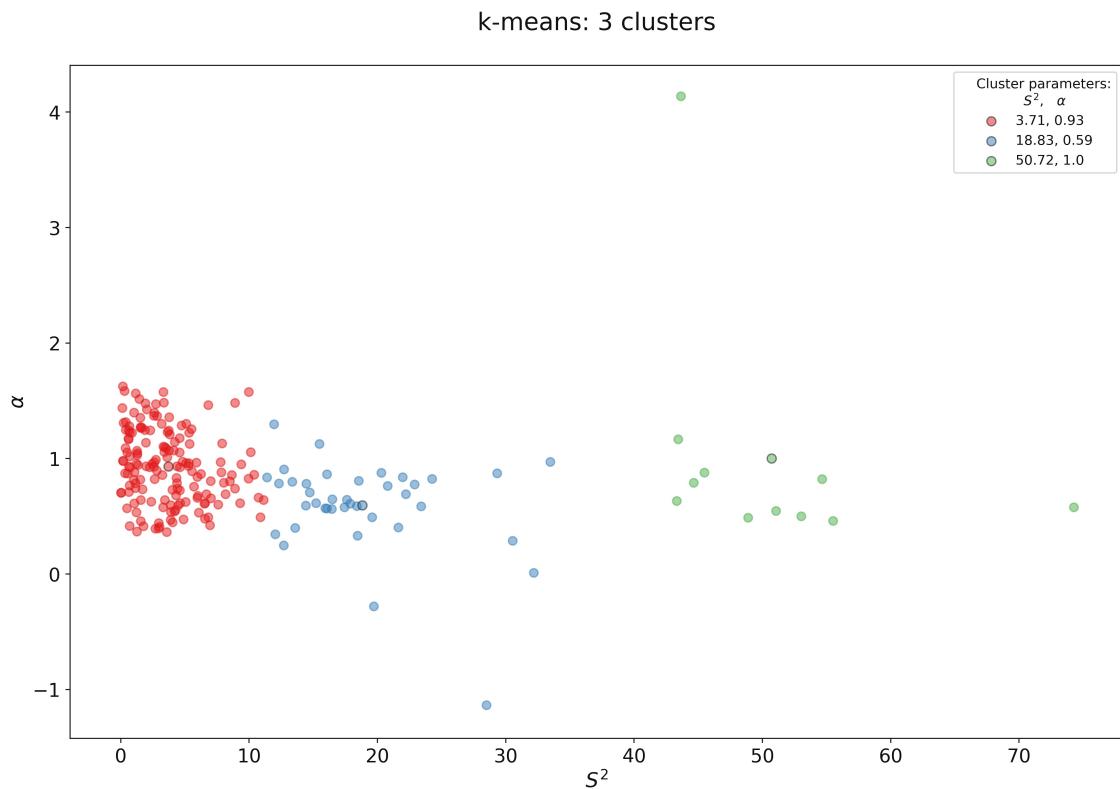


Figura 6.3.1: Resultado do agrupamento do número de casos diários de Covid19 para todos os países no espaço  $S^2 \times \alpha$ .

Nesse exercício, os grupos de países no espaço  $S^2 \times \alpha$  se encontram nos seguintes arquivos: *Exercicio6\_3\_S2vsAlfa\_cluster\_0\_x\_3.67\_y\_0.93\_flags.csv* (cluster vermelho), *Exercicio6\_3\_S2vsAlfa\_cluster\_1\_x\_18.64\_y\_0.66\_flags.csv* (cluster azul) e *Exercicio6\_3\_S2vsAlfa\_cluster\_2\_x\_53.82\_y\_1.04\_flags.csv* (cluster verde).

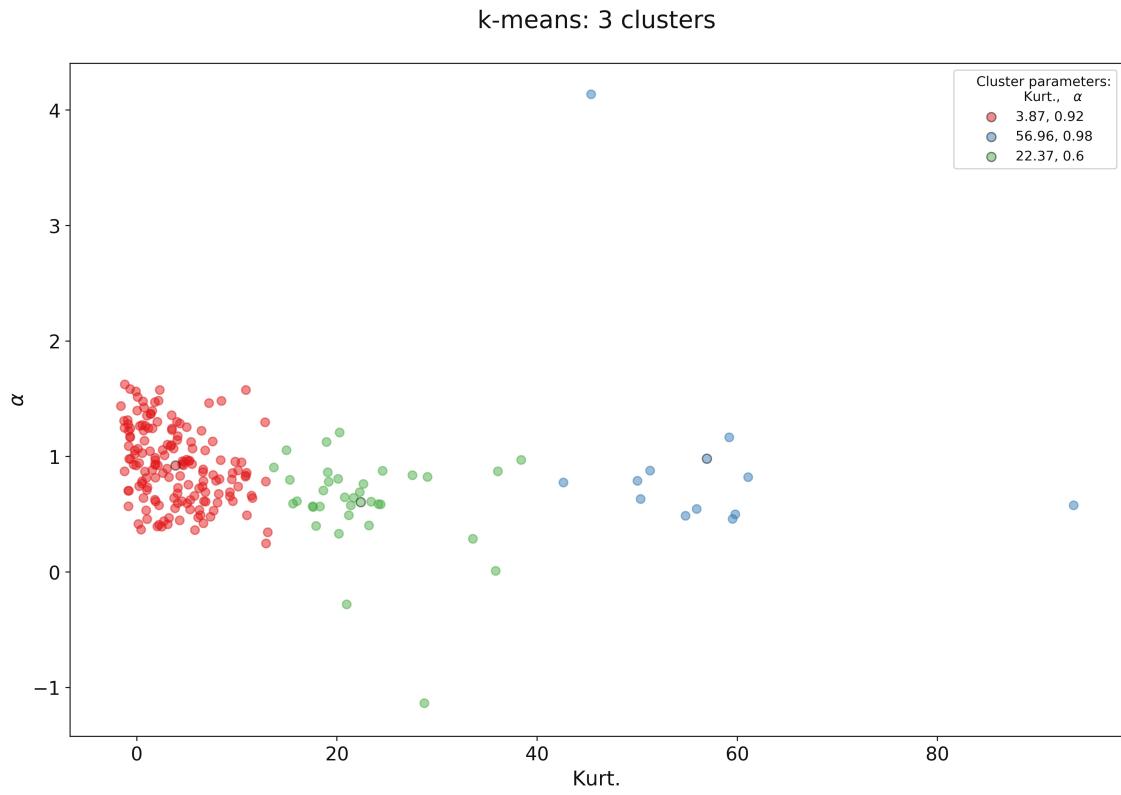


Figura 6.3.2: Resultado do agrupamento do número de casos diários de Covid19 para todos os países no espaço  $K \times \alpha$ .

Nesse exercício, os grupos de países no espaço  $K \times \alpha$  se encontram nos seguintes arquivos: *Exercicio6\_3\_KvsAlfa\_cluster\_0\_x\_3.61\_y\_0.92\_flags.csv* (cluster vermelho), *Exercicio6\_3\_KvsAlfa\_cluster\_1\_x\_58.83\_y\_1.02\_flags.csv* (cluster verde) e *Exercicio6\_3\_KvsAlfa\_cluster\_2\_x\_20.72\_y\_0.66\_flags.csv* (cluster azul).

## Exercício 7 - Singularity Multifractal Spectra (SMS), também conhecido como MDFA

Os resultado das análises das famílias de sinais deste exercício se encontram na pasta **Exercise7** organizados nas pastas **7.2** e **7.3**.

### 7.1

O algoritmo foi alterado e se encontra na pasta **statistical\_analysis\_codes**.

### 7.2

Os códigos desta questão se encontram separados pelos grupos da tabela *Dataset\_signal*. Os resultados a seguir são para o grupo noise com  $n = 8192$ , e estão presentes na pasta **grupo\_noise** dentro da pasta referente a este exercício - **7.2**.

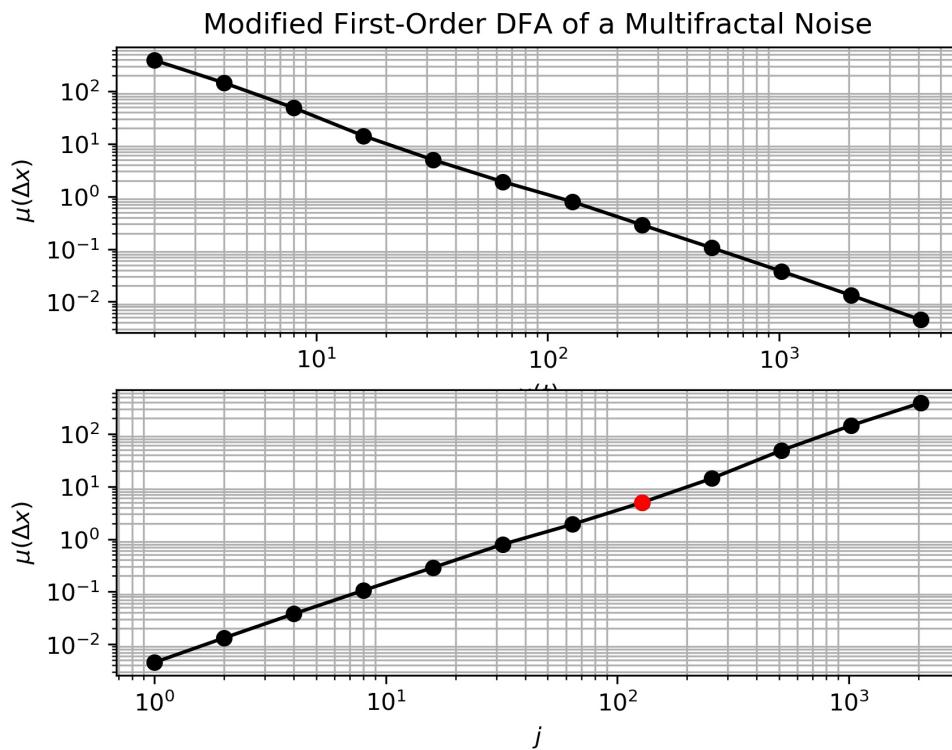


Figura 7.2.1.: Topo: média da função de flutuação x tempo medido nas diferentes escalas; abaixo: média da função de flutuação x escala. Ambos os plots em log-log. Grupo noise com  $n = 8192$ .

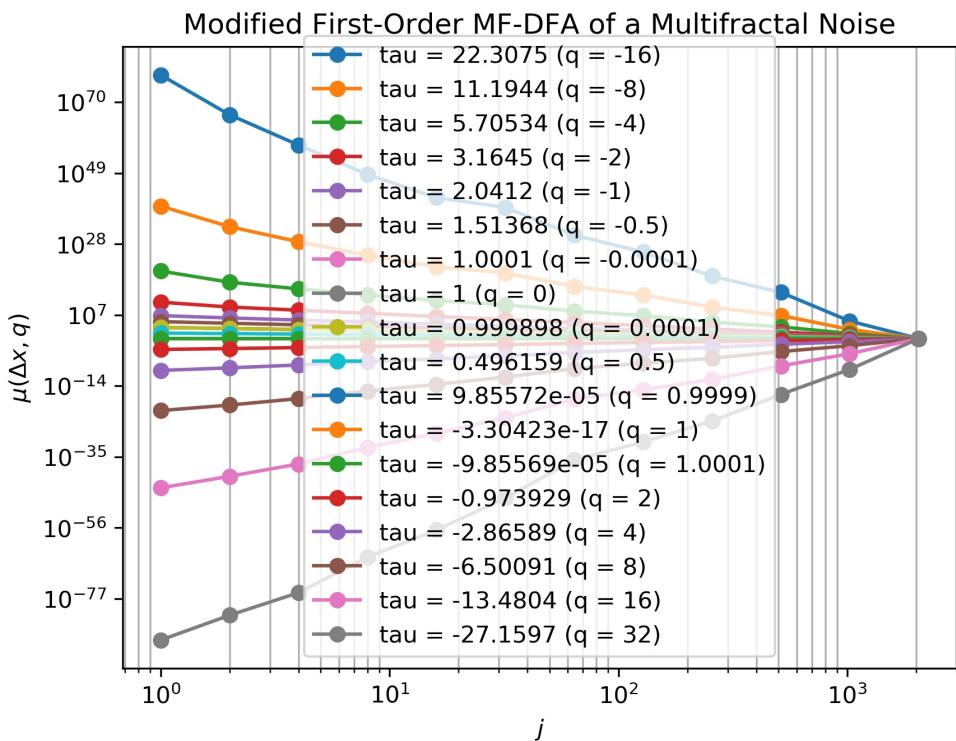


Figura 7.2.2: Função de flutuação x escala em plot log-log para diferentes valores do expoente  $q$ . Grupo noise com  $n = 8192$ .



Figura 7.2.3: Dependência de  $\tau(q)$  com  $q$  do grupo noise para  $n = 8192$ .

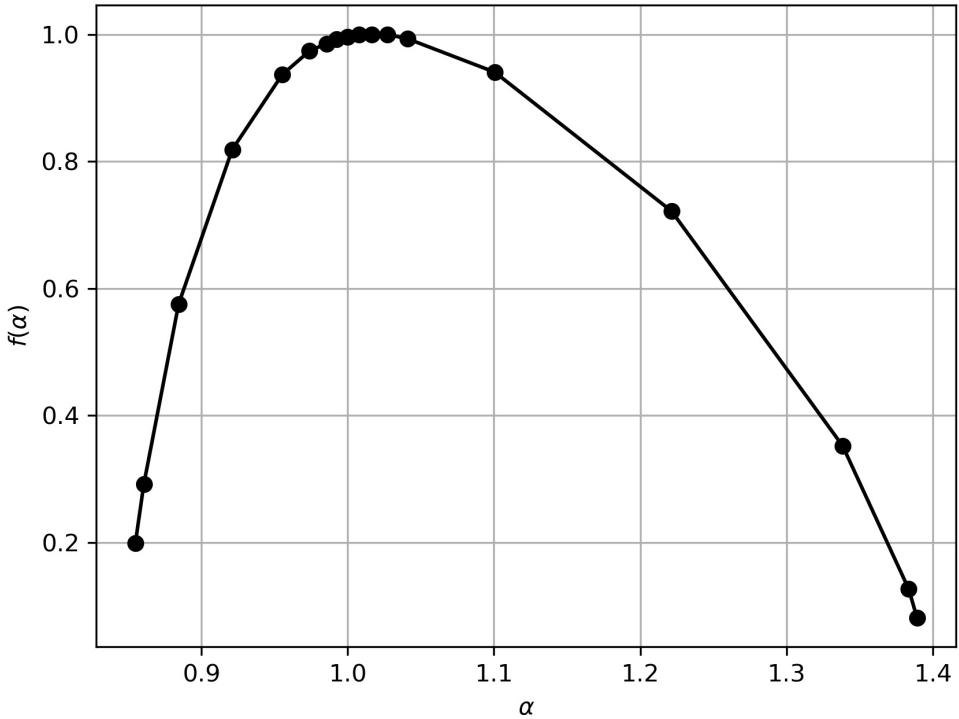


Figura 7.2.4: Espectro de singularidade  $f(\alpha)$  vs  $\alpha$  do grupo noise com  $n = 8192$ .

### Monofractalidade vs Multifractalidade

Uma análise interessante é a comparação do espectro de singularidade entre dois grupos particulares da tabela *Dataset\_signal*: *chaosnoise* e *pmnoise*. O espectro de singularidade para o mapeamento Logístico e para uma série exógena (Figura 7.2.5) não apresentados a seguir.

O valor de  $\Delta\alpha = \alpha_{max} - \alpha_{min}$  reflete a multifractalidade da série. Quanto maior o  $\Delta\alpha$ , maior a característica multifractal e complexidade da série, e a distribuição apresenta flutuações mais severas. Mapas caóticos como o mapeamento Logístico possuem uma faixa dinâmica estreita e apresentam monofractalidade, de modo que seu espectro de singularidade indica um valor de  $\Delta\alpha$  relativamente pequeno: a curva em vermelho da Figura 7.2.5 não se estende nas duas extremidades (ela é monofractal-like). Por outro lado, a série exógena do pmodel apresenta multifractalidade, de forma que seu  $\Delta\alpha$  é maior conforme evidenciado pelo espectro de singularidade.

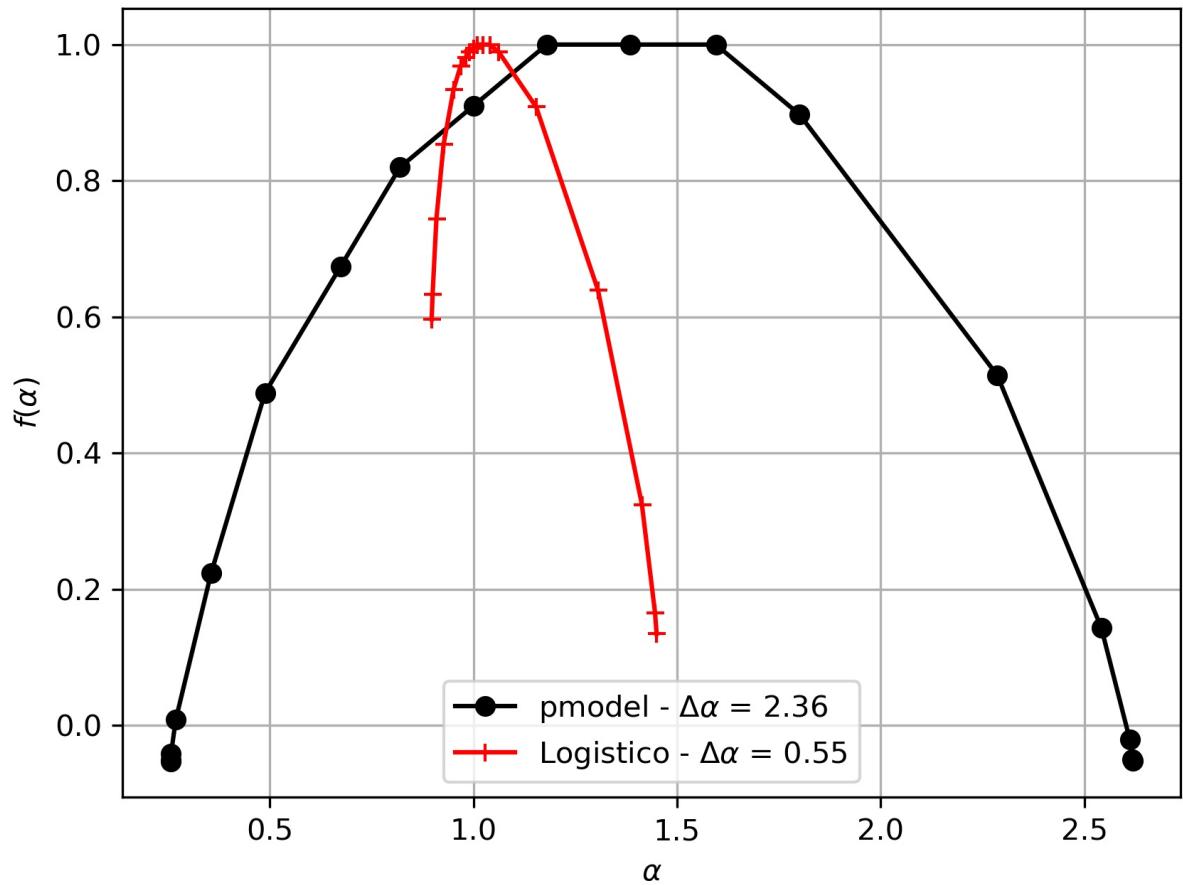


Figura 7.2.5: Comparaçāo entre o espectro de singularidade  $f(\alpha)$  vs  $\alpha$  do grupo pmnoise e do grupo chaosnoise. Série exógena do pmodel com  $p = 0.18$  e  $\beta = 0.7$  vs mapeamento Logístico com  $\rho = 3.81$ .

### 7.3

Os resultados deste exercício estão presentes na pasta **7.3**. Abaixo o plot do melhor número de clusters determinado pelo méotodo do cotovelo. O algoritmo utilizado foi o kmeans\_2D\_group\_flags.py, que agrupou cada país em seu respectivo cluster em arquivos .csv (identificados pelo centróide do grupo).

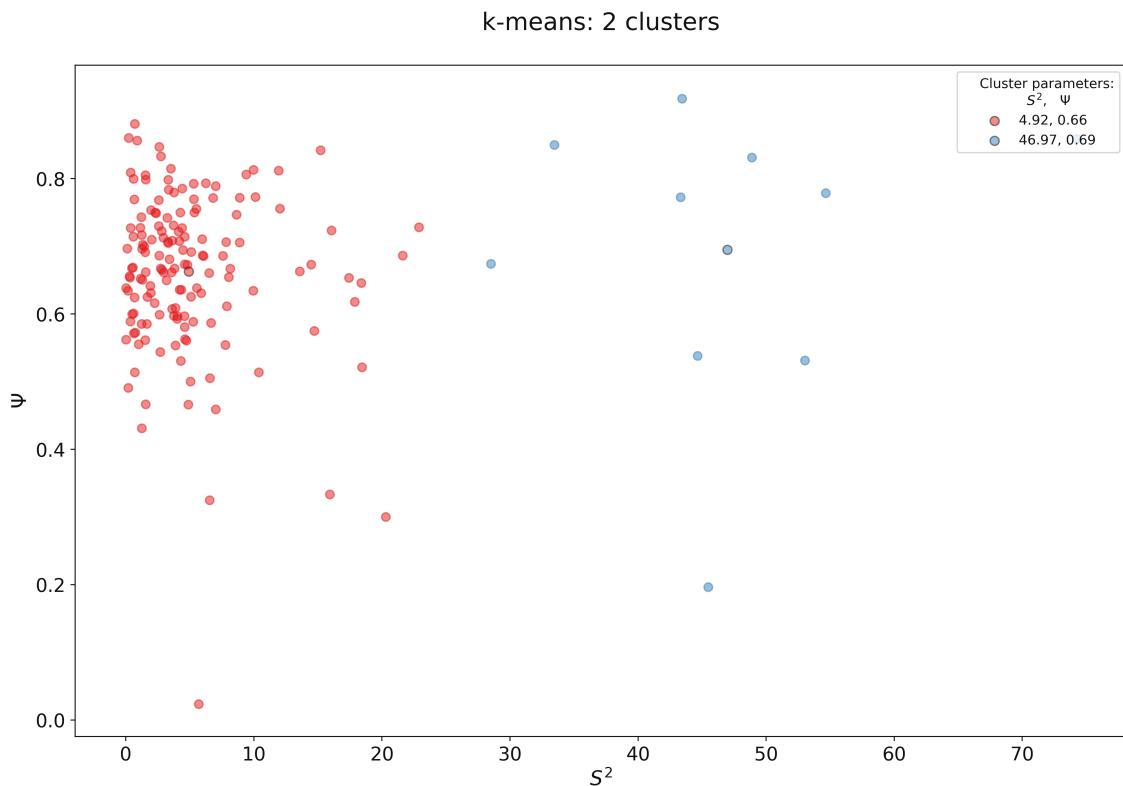


Figura 7.3.1: Resultado do agrupamento do número de casos diários de Covid19 para todos os países no espaço  $S^2$  x  $\Psi$ . O valor  $n\_c = 2$  obteve a melhor performance.

Neste exercício, os grupos de países se encontram nos seguintes arquivos: *Exercicio7\_3\_cluster\_0\_x\_3.58\_y\_0.62\_flags.csv* (cluster vermelho) e *Exercicio7\_3\_cluster\_1\_x\_36.38\_y\_0.62\_flags.csv* (cluster azul).

## Exercício 8 - Global Wavelet Spectrum

Os resultados deste exercício se encontram na pasta **Exercise8**. Abaixo está a análise para a série de novos casos da Covid19 para o Brasil, de 30 de março a 5 de junho. Foi utilizado o pacote do Python waipy.

Os plots das ondeletas mãe Morlet e DoG (Derivative of Gaussian) indicam que a melhor escolha é a DoG, pois ela foi capaz de captar os períodos do sinal dentro do cone de influência, com um pico bem pronunciado no wavelet spectrum (Figura 8.2.1). Por outro lado, a ondeleta Morlet não foi capaz de apresentar o pico dentro do cone de influência (Figura 8.1.1).

### 8.1

Resultado da aplicação do pacote waipy com a wavelet mãe Morlet, aplicada sobre a série da Covid19 (novos casos diários) para o Brasil. O período do sinal não é bem captado pela ondeleta: o pico do espectro wavelet - em azul à direita - está fora do cone de influência.

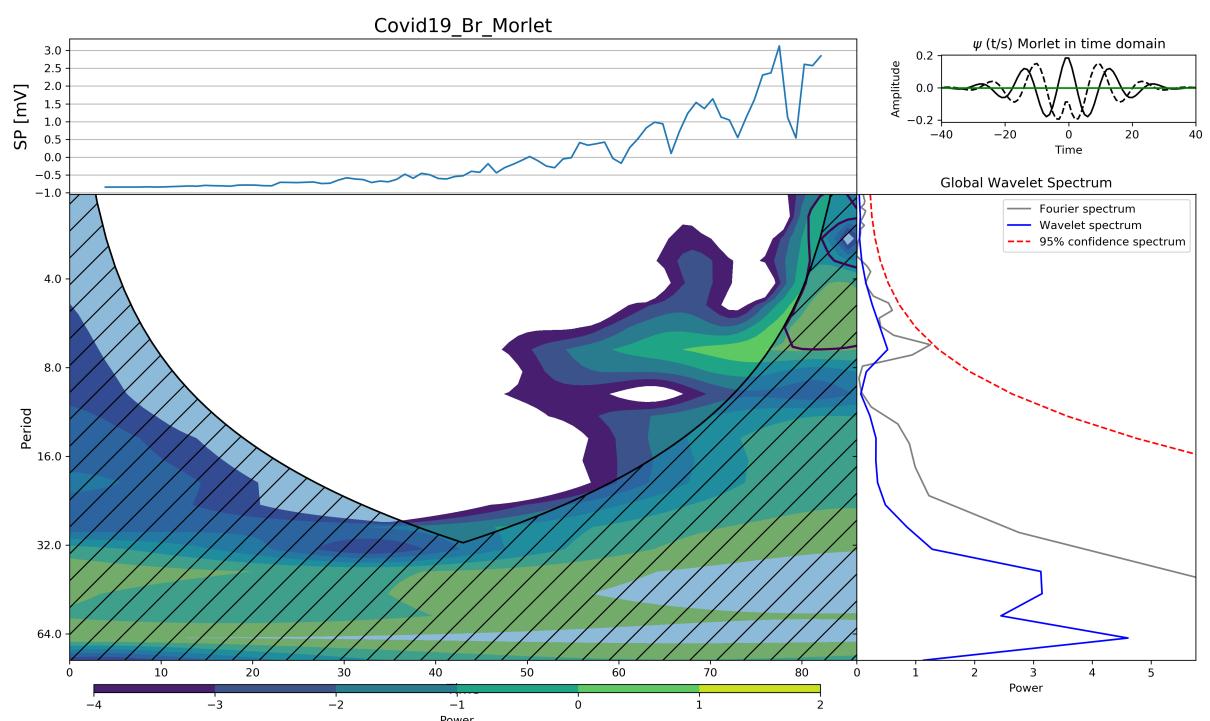


Figura 8.1.1: Global Wavelet Spectrum da série de dados da Covid19 do Brasil com a ondeleta mãe Morlet.

## 8.2

Resultado da aplicação do waipy com a wavelet mãe DoG, aplicada sobre a série da Covid19 (novos casos diários) para o Brasil. Parte do pico do espectro wavelet - em azul à direita - está dentro do cone de influência.

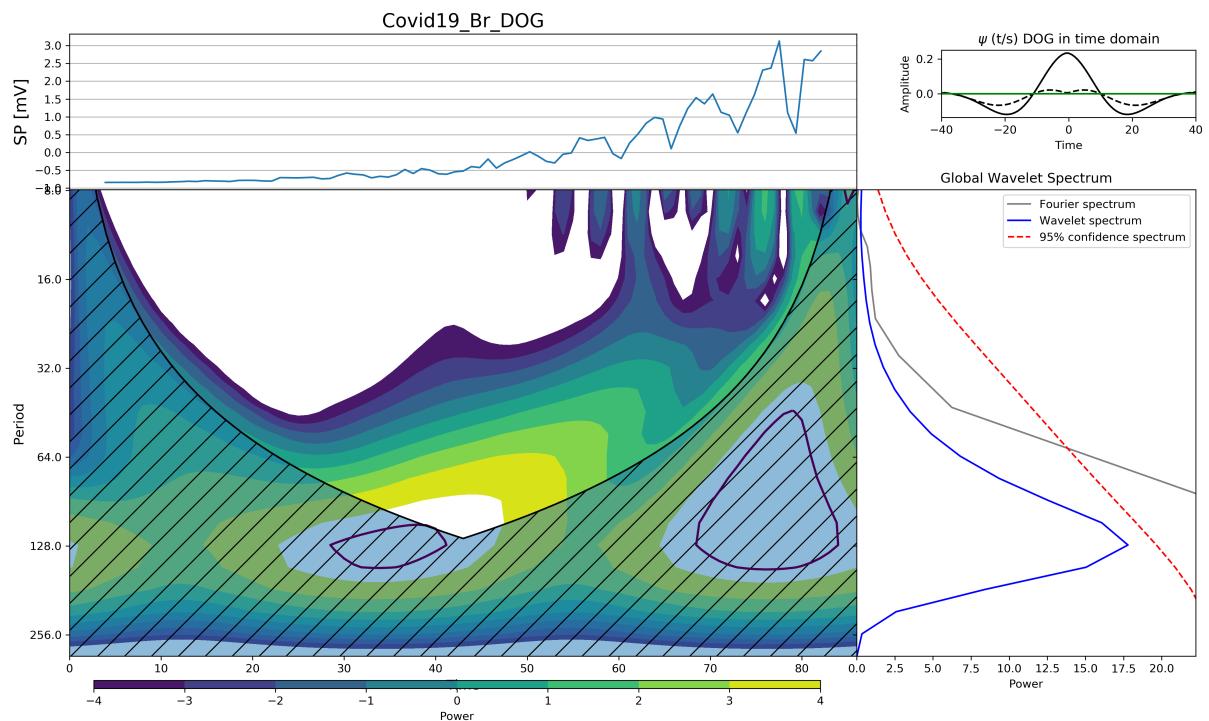


Figura 8.2.1: Global Wavelet Spectrum da série de dados da Covid19 do Brasil com a ondeleta mãe DoG (Derivative of Gaussian).

## Exercício 9 - Self-Organized Criticality (SOC)

Os resultados deste exercício se encontram na pasta **Exercise9**, organizados nas pastas **9.1** e **9.2**.

### 9.1

O resultado a seguir resume a aplicação do algoritmo SOC.py às séries endógenas e exógenas do p-model. Pode-se observar que as séries endógenas apresentam Self Organized Criticality.

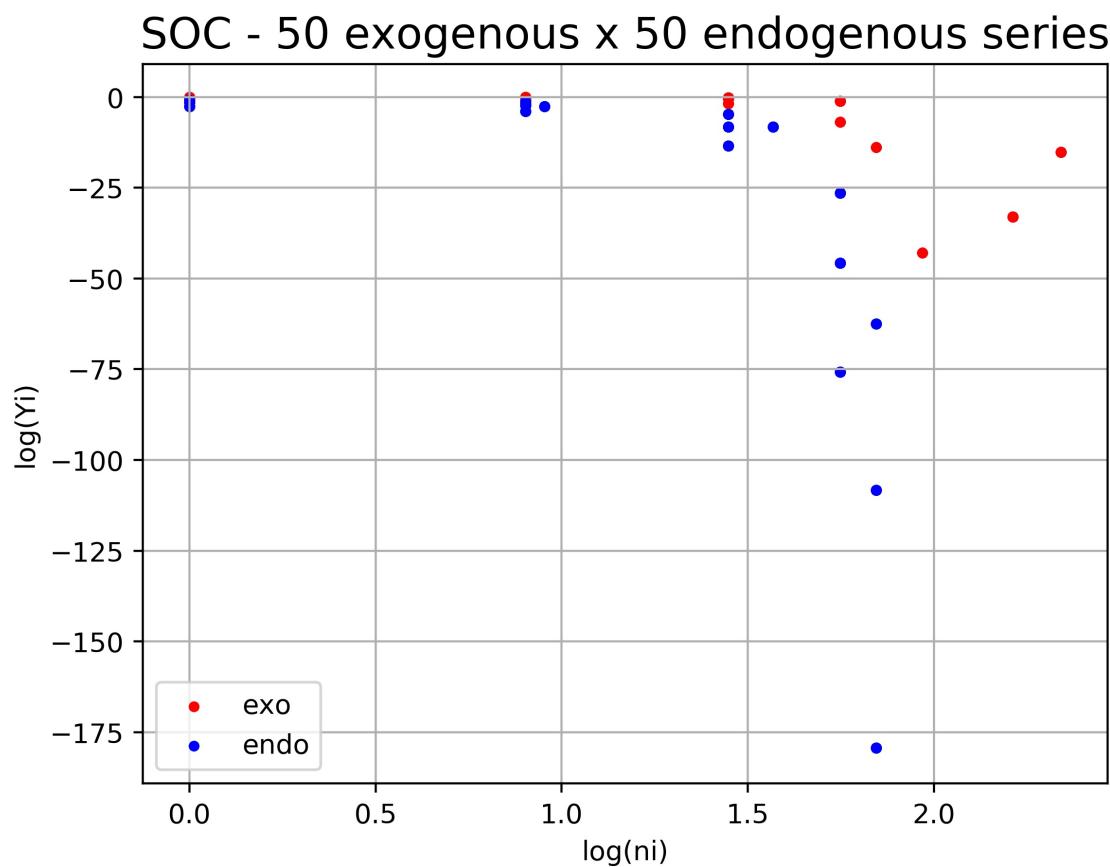


Figura 9.1.1: Plots de Self Organized Criticality para 50 sinais endógenos (em azul) e 50 sinais exógenos (em vermelho). Observa-se que as séries endógenas apresentam perfil de Self Organized Criticality, enquanto as exógenas não.

## 9.2

Os plots resultantes da análise de todos os países da série *ST-OWS\_NDC\_Covid19* se encontram na pasta **paises** (dentro da pasta referente a este exercício). A seguir é apresentado o plot do resultado para todos os países da série juntos. O espalhamento de pontos na parte superior do gráfico pode ser interpretado como a presença de diferentes pontos de relaxação para a série, bem como diferentes leis de potência. Além disso, a parte inferior apresenta uma cauda, evidenciando um comportamento de relaxação em um ponto crítico apenas. Ou seja, a menos dos pontos na parte superior do gráfico, há uma assinatura única mais proeminente que pode caracterizar Self Organized Criticality.

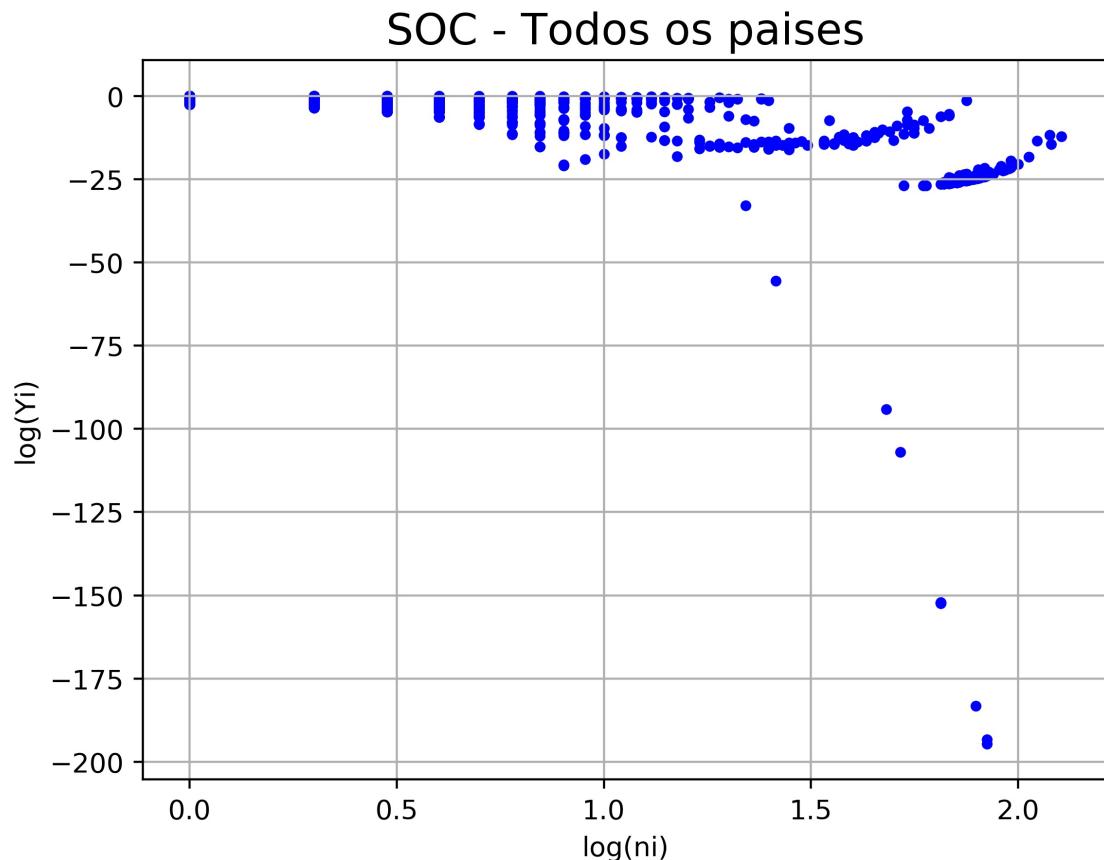


Figura 9.2.1: Plot de todos os países usando o algoritmo SOC.py.