INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS - INPE
PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
MATEMÁTICA COMPUTACIONAL I (CAP-239)

LISTA DE EXERCÍCIOS
ANÁLISE ESPECTRAL DE PROCESSOS ESTOCÁSTICOS

WILLIAN VIEIRA DE OLIVEIRA, 142182

SÃO JOSÉ DOS CAMPOS - SP
2019

Observação: Os algoritmos utilizados para responder esta lista de exercícios estão disponíveis a partir do seguinte endereço:

- https://github.com/wivoliveira/CAP-239

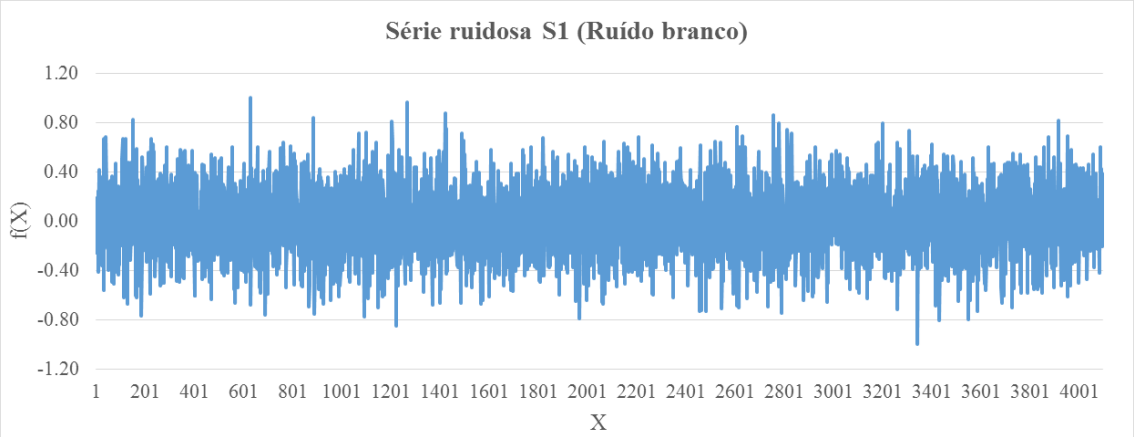
---

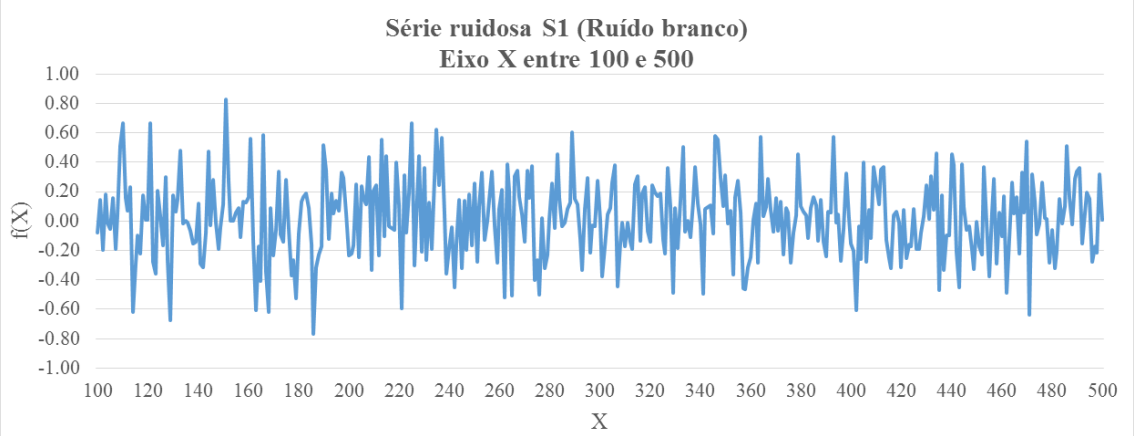
1. Simulação de Sinais Estocásticos com $N = 2^{12}$ valores de medidas.

1.1. Utilize o algoritmo *powernoise.m* e gere os seguintes ruídos $1/f^{\beta}$:

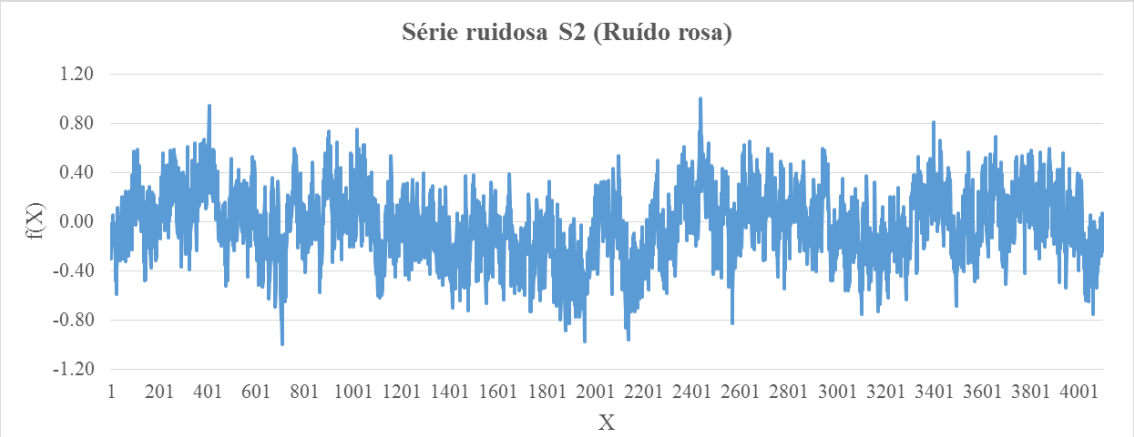
Algoritmo *powernoise.m* (Matlab): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%201/powernoise.m

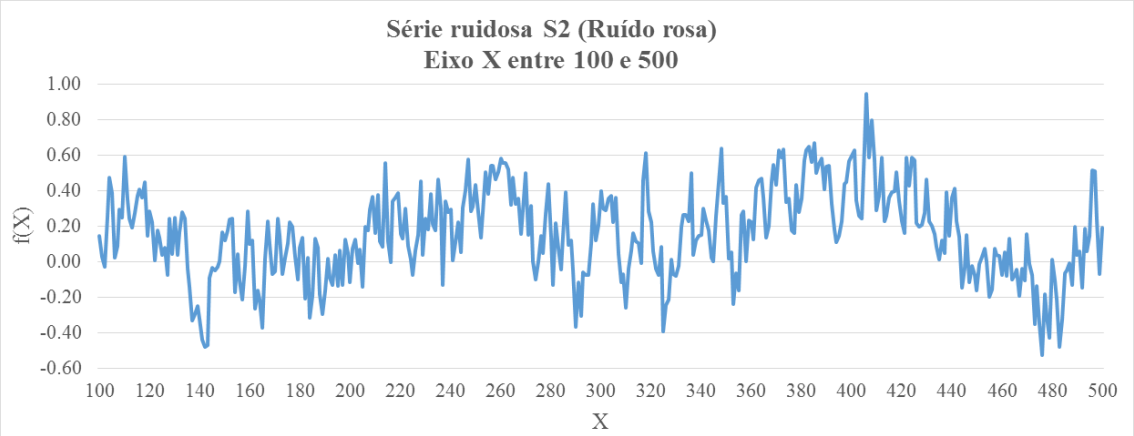
$S1$: $\beta = 0$ (white noise)



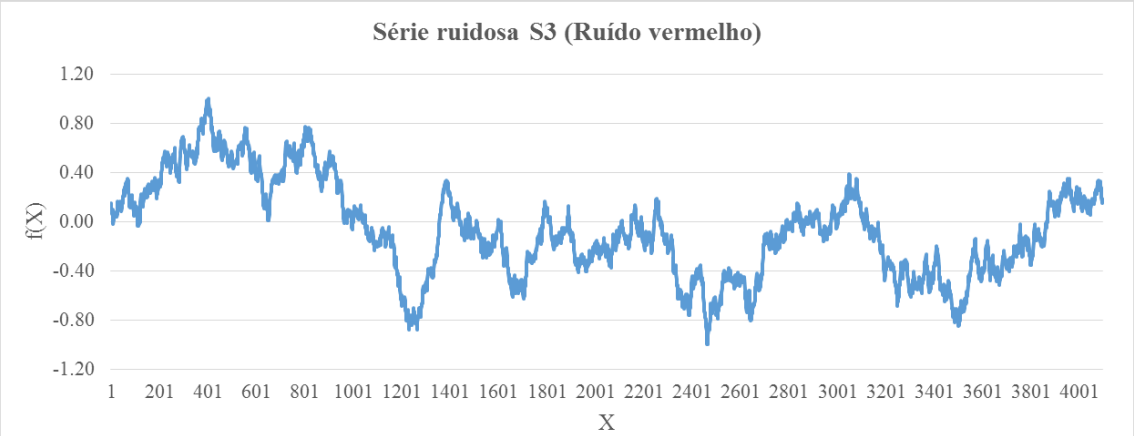


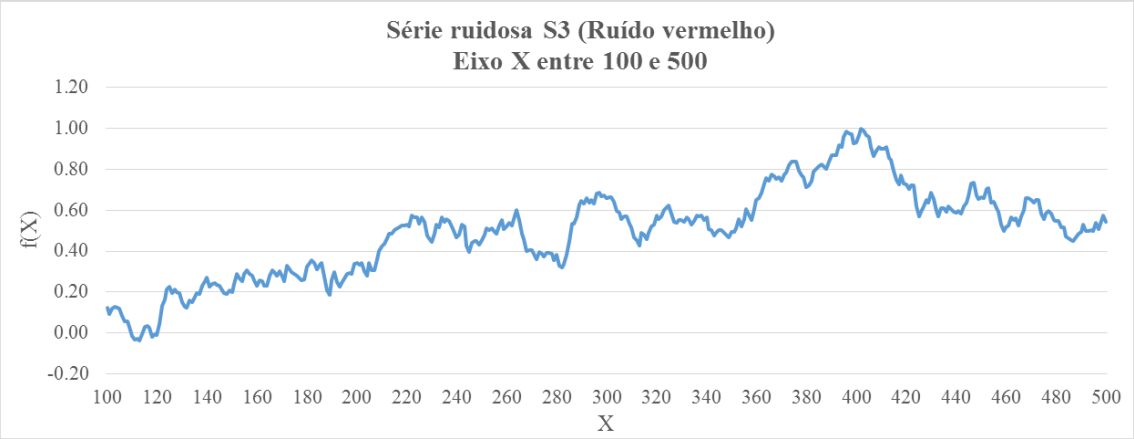
$S2$: $\beta = 1$ (pink noise)





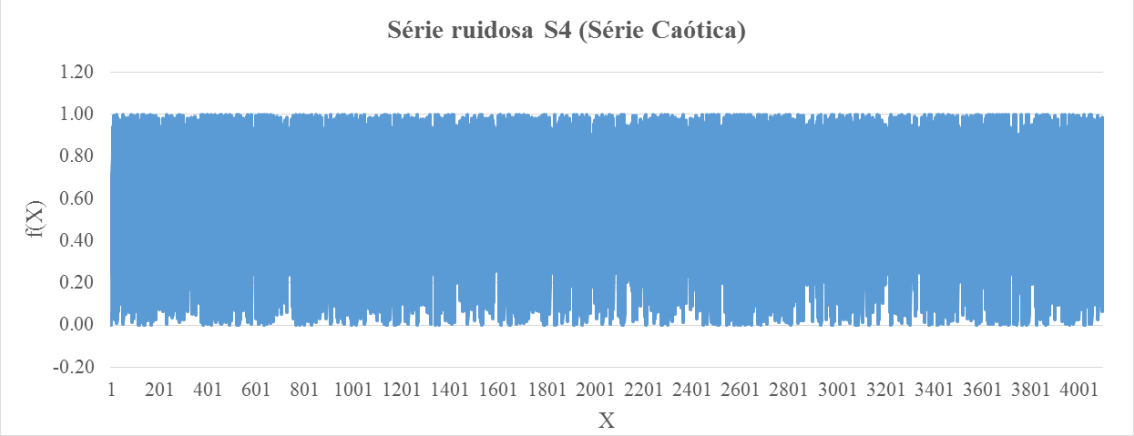
$S3$: $\beta = 2$ (red noise)

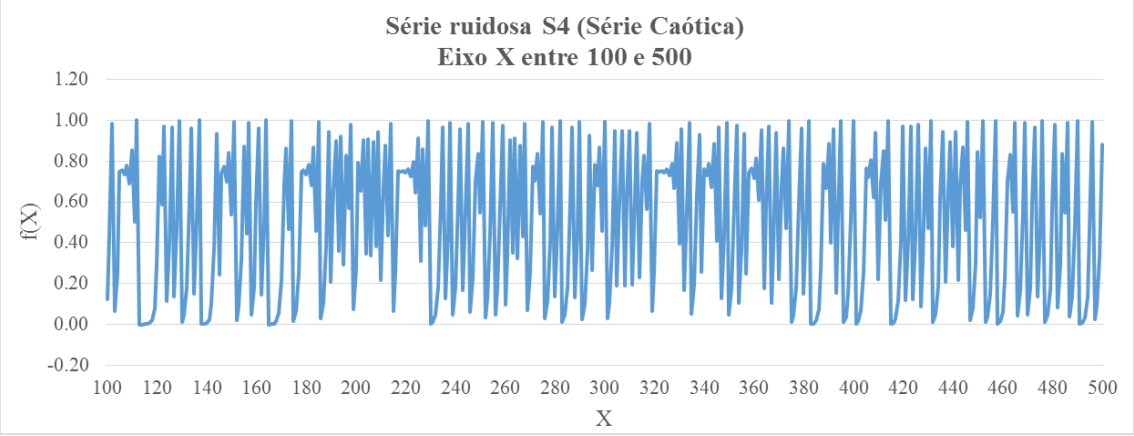




1.2. *S4*: Série Caótica gerada a partir do Mapeamento Quadrático (Logístico) para $\rho = 4$, considerando $A_0 = 0.001$.

Algoritmo *Serie_caotica.py* (Python): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%201/Serie_caotica.py

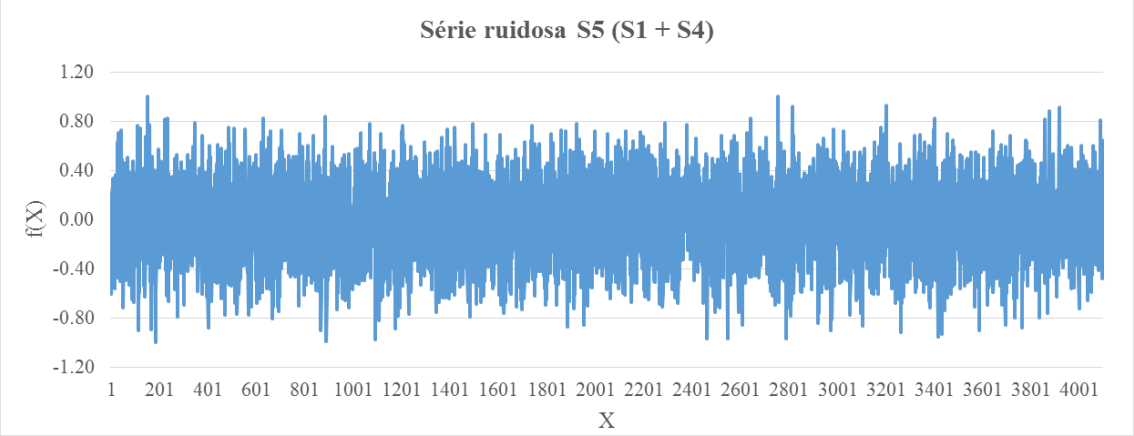


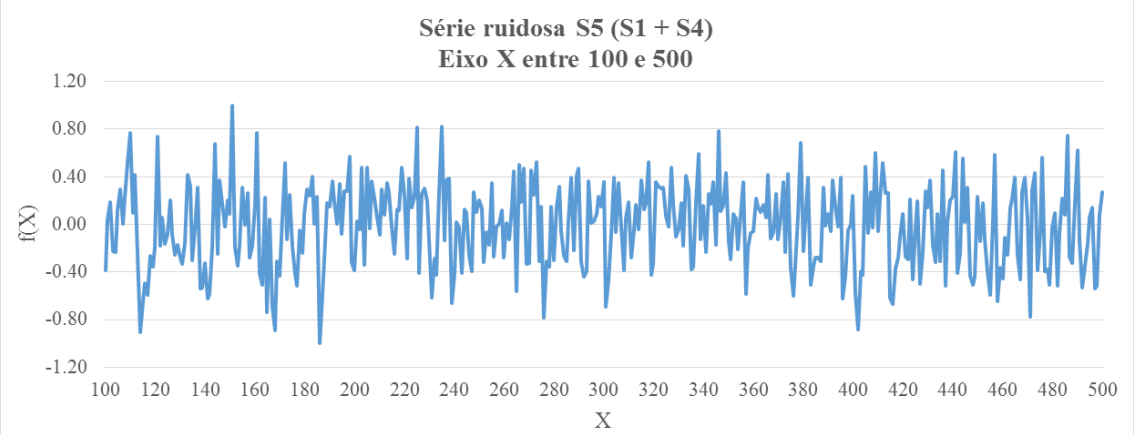


1.3. Gere um algoritmo em Python para somar e normalizar com <A> = 0, dois sinais com o mesmo tamanho e construa as séries:

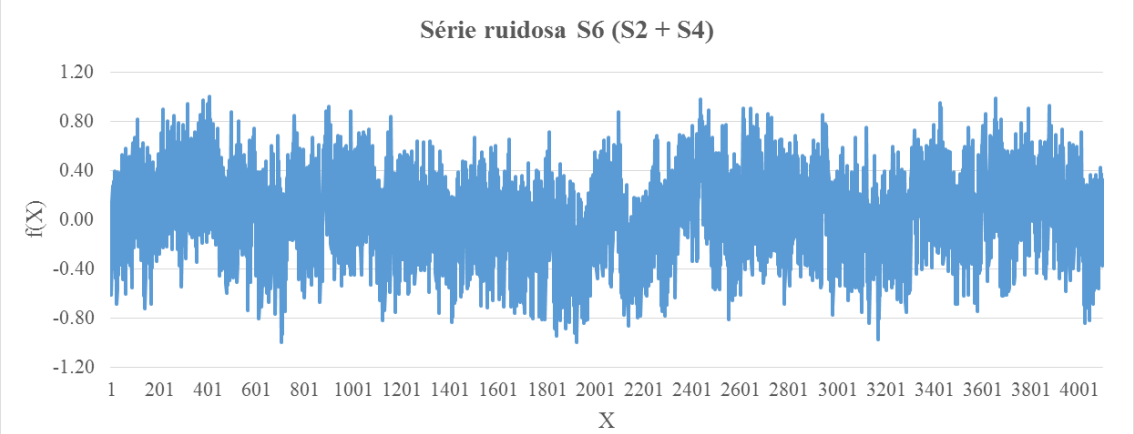
Algoritmo *Somar_Normalizar.py* (Python): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%201/Somar_Normalizar.py

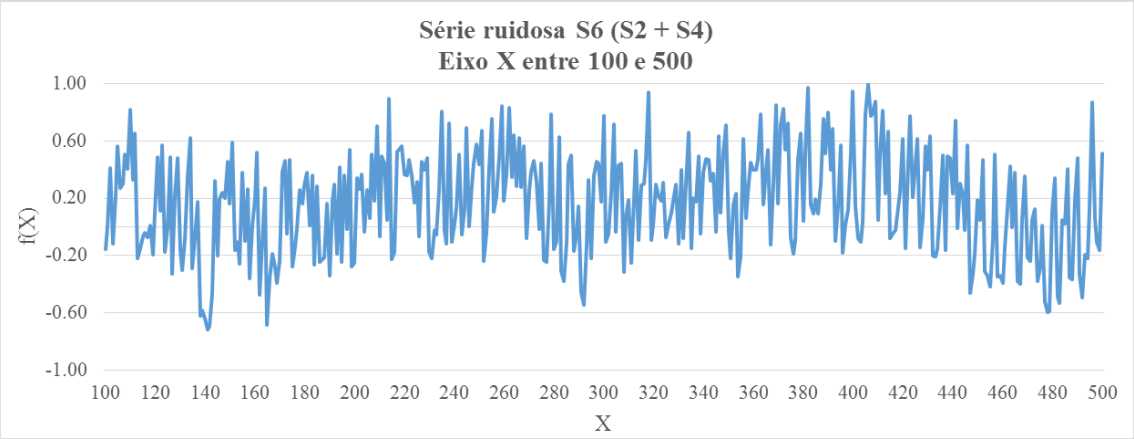
$\boldsymbol{S5: S1 + S4}$



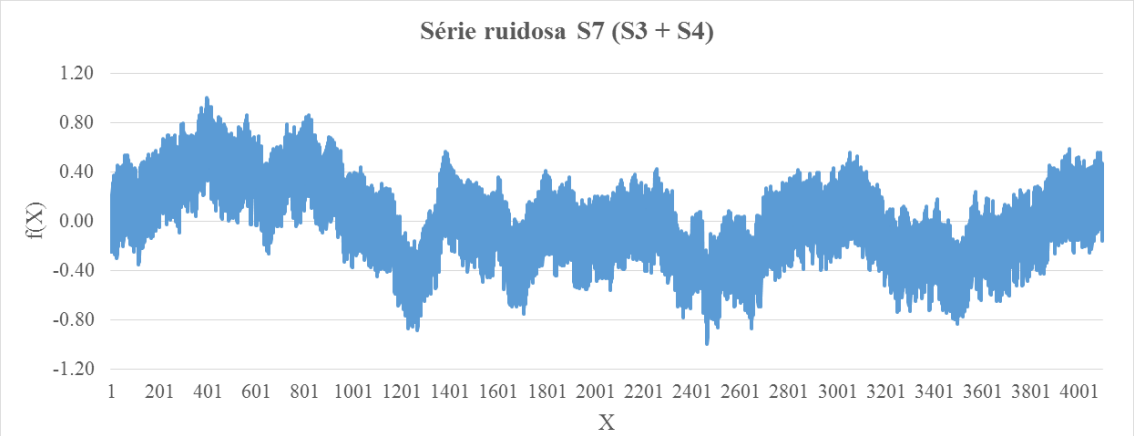


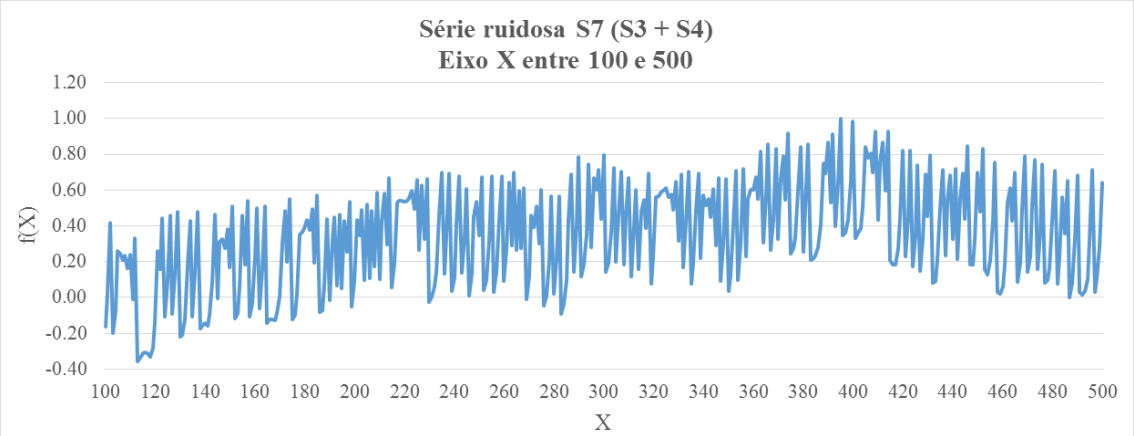
$\boldsymbol{S6: S2 + S4}$





S7*: *S3* + *S4



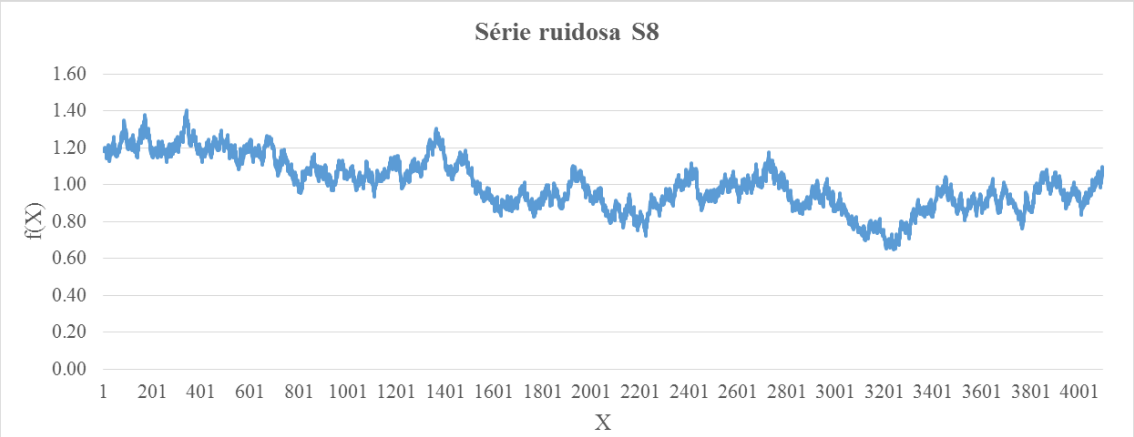


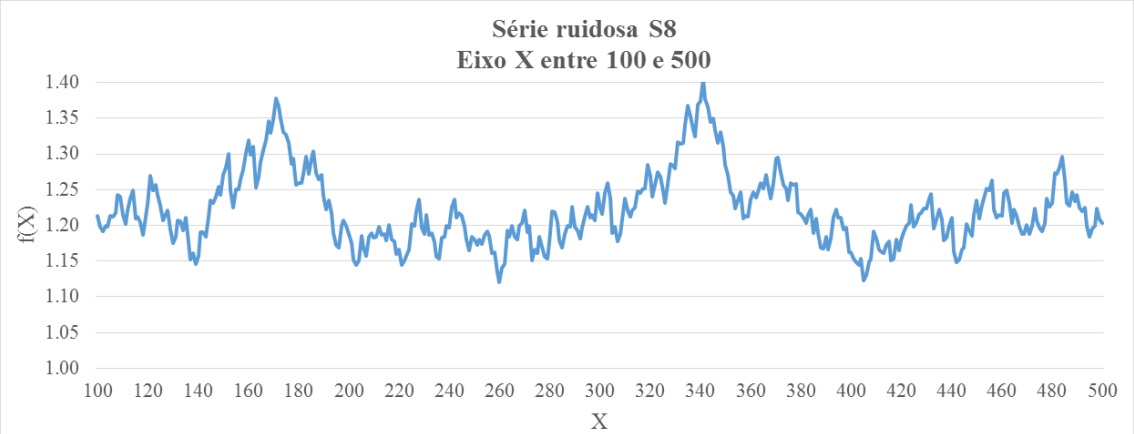
1.4. Utilize o algoritmo *pmodel.m* e gere os seguintes sinais com $N = 2^{12}$ valores de medidas:

Algoritmos em Matlab e Python (**Exercício Extra!**):

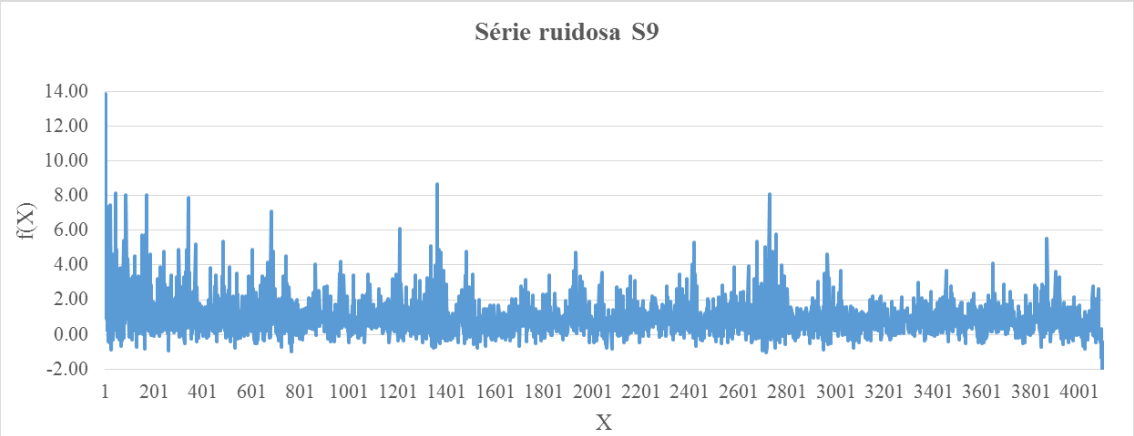
- https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%201/pmodel.m
- https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%201/pmodel.py

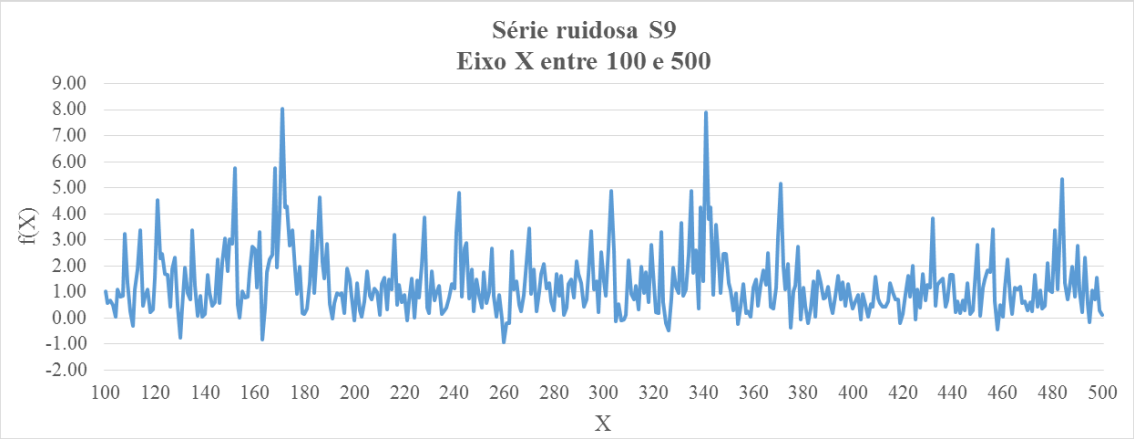
$$S8: \rho = 0.52;\ \beta = -1.66$$



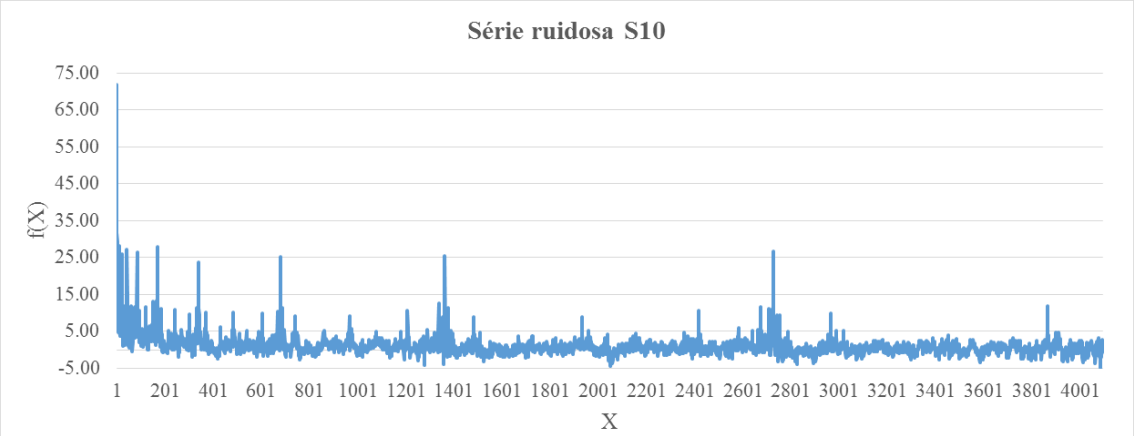


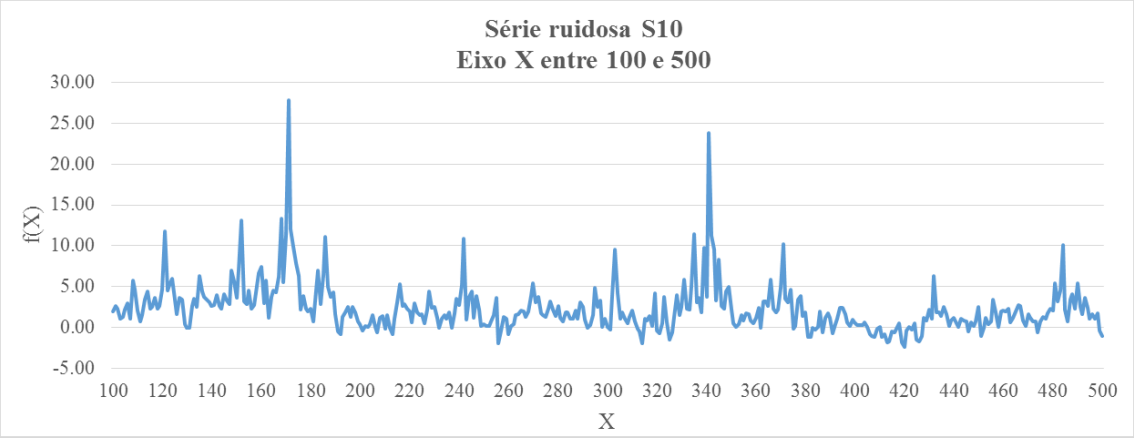
$$S9: \rho = 0.62;\ \beta = -0.45$$





$$\boldsymbol{S10: \rho = 0.72;\ \beta = -0.75}$$





2. Distribuições de Probabilidades:

2.1. Escreva um programa em Python que, para uma amostra com N resultados, ajuste as seguintes PDFs: (i) Uniforme, (ii) Binomial, (iii) Beta, (iv) Laplace,

(v) Gamma, (vi) Expoencial, (vii) Qui-quadrado, (viii) Cauchy e (ix) Gaussiana (Normal).

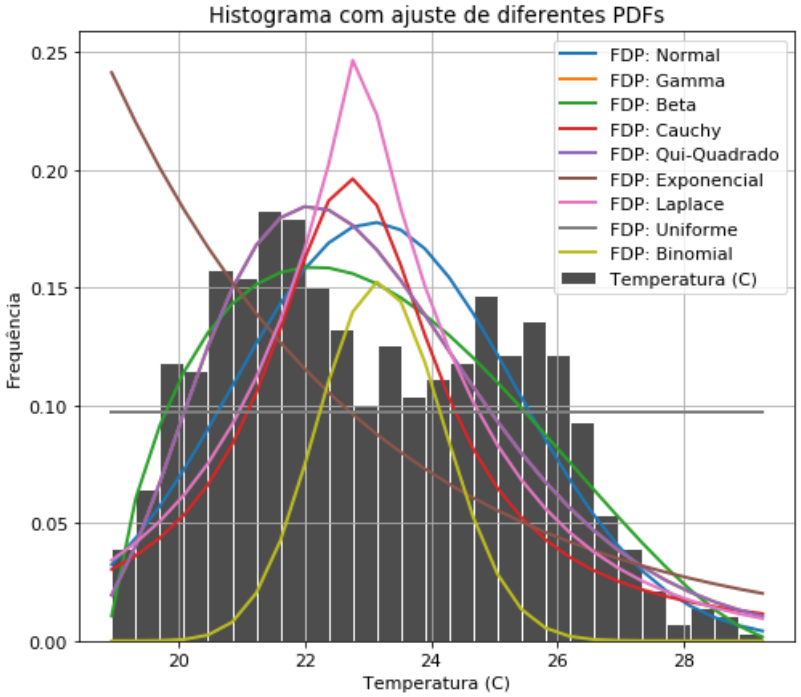
O algoritmo gerado para responder esta questão (*FittingDistributions.py*) está disponível em: <https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%202/FittingDistributions.py>.

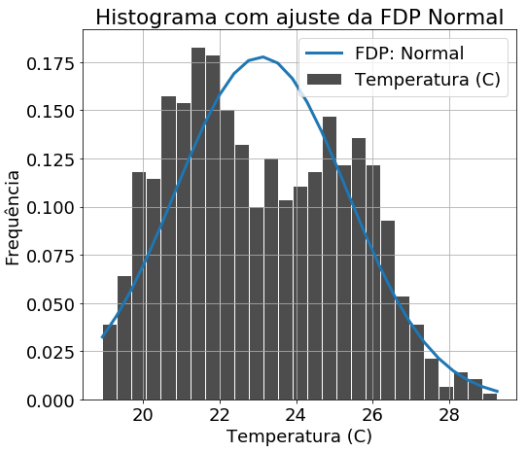
Exemplo implementado a partir do código gerado:

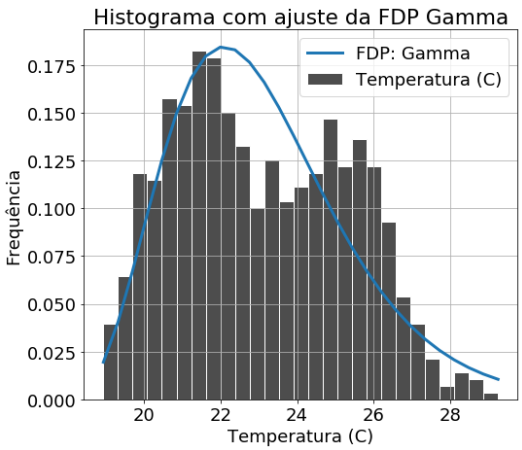
- Dados: valores de temperatura da superfície do mar disponíveis acessados diretamente pela biblioteca ‘pandas’.

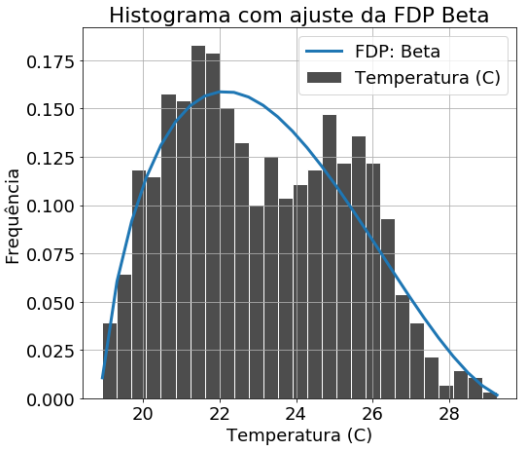
  Fonte: http://www.statsmodels.org/dev/datasets/generated/elnino.html

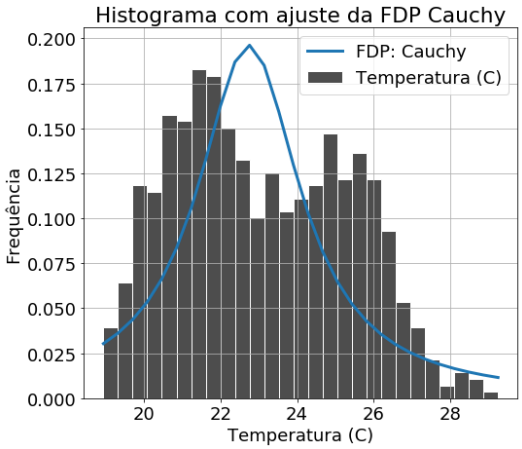
- Bins: 27

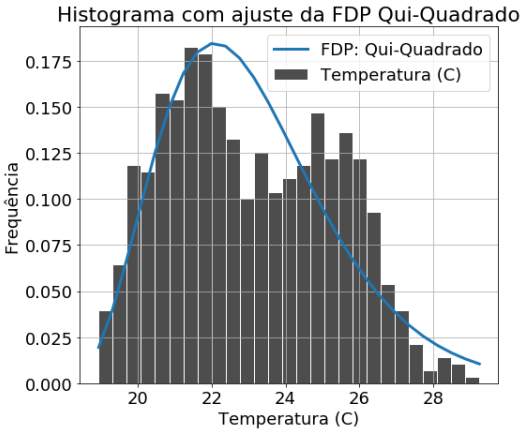


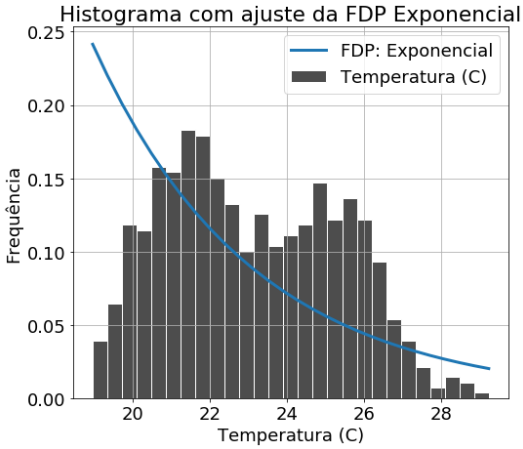


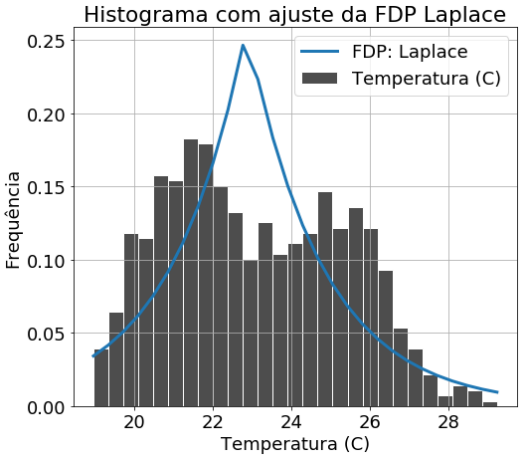


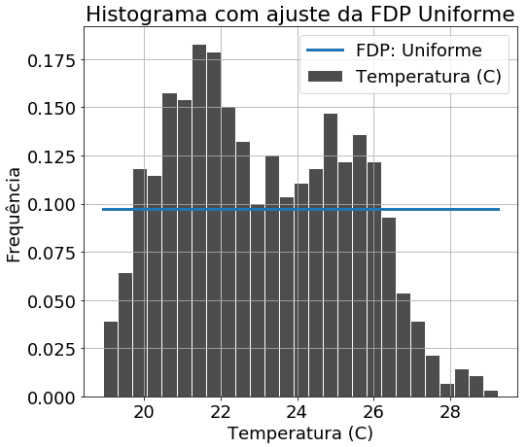

Histograma com ajuste da FDP Beta
FDP: Beta
Temperatura (C)
Frequência
Temperatura (C)

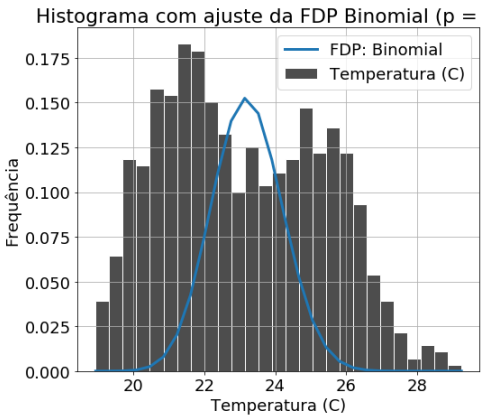

Histograma com ajuste da FDP Cauchy
FDP: Cauchy
Temperatura (C)
Frequência
Temperatura (C)


Histograma com ajuste da FDP Qui-Quadrado
FDP: Qui-Quadrado
Temperatura (C)
Frequência
Temperatura (C)


Histograma com ajuste da FDP Exponencial
FDP: Exponencial
Temperatura (C)
Frequência
Temperatura (C)


Histograma com ajuste da FDP Laplace
FDP: Laplace
Temperatura (C)
Frequência
Temperatura (C)

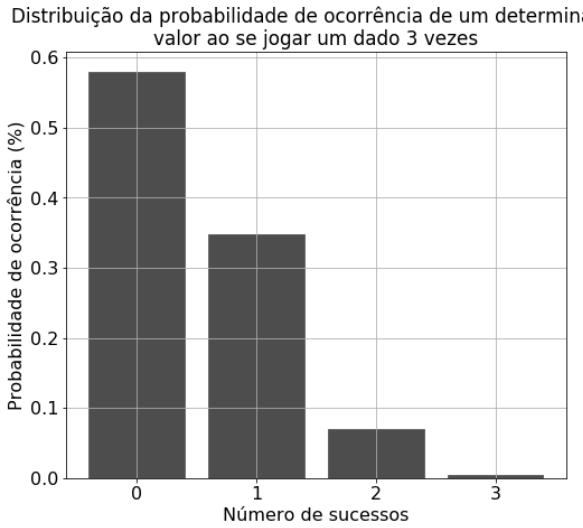

Histograma com ajuste da FDP Uniforme
FDP: Uniforme
Temperatura (C)
Frequência
Temperatura (C)

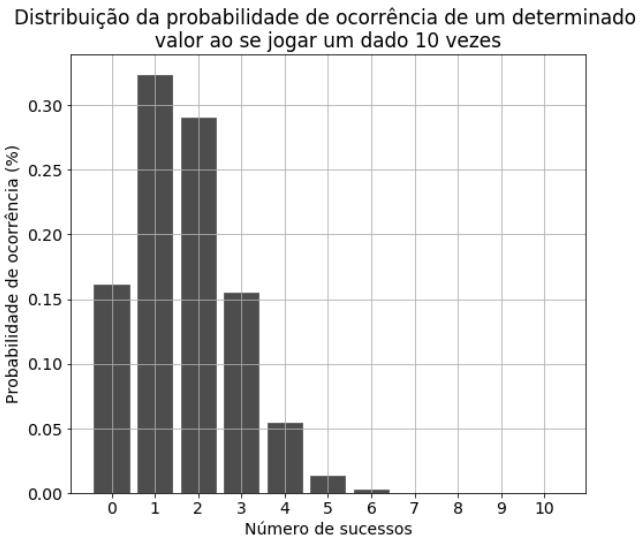

Histograma com ajuste da FDP Binomial (p = 0.4)
FDP: Binomial
Temperatura (C)
Frequência
Temperatura (C)

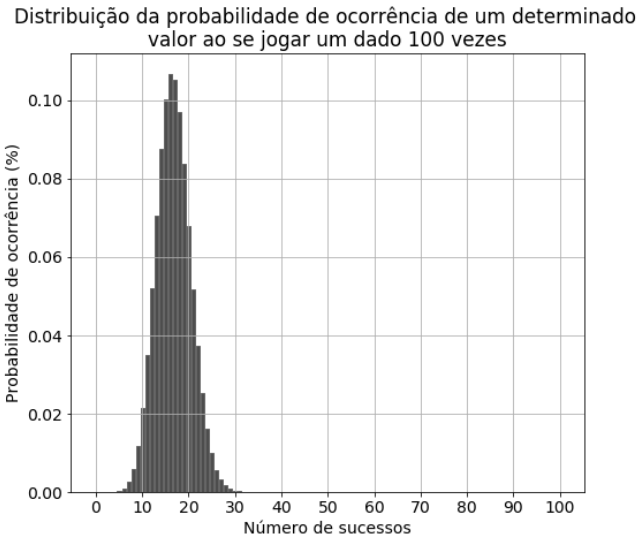
2.2. Considere o seguinte experimento: Lançamento de 3 dados simultâneos com registro de quantas vezes um determinado resultado pode ser obtido. Mostre que a distribuição limite é binomial e que com N tendendo a infinito ela converge para uma Gaussiana.

Algoritmo gerado (Dados_simultaneos.py): <https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%202/Dados_simultaneos.py>.

Este efeito é descrito pelo Teorema do Limite Central, o qual estabelece que a medida que se aumenta o tamanho da amostra, a distribuição binomial se aproxima de uma distribuição gaussiana (normal). Este efeito é exemplificado nos gráficos abaixo. Observe que, ao se aumentar o número de dados lançados (e.g., 3, 10 e 100 dados), a distribuição das probabilidades de ocorrência de um determinado valor ser aproxima de uma distribuição normal.







3. Probabilidade Condicional:

3.1. Considere 3 regiões do céu contendo aproximadamente o mesmo número (N) de galáxias, cujas Distribuições morfológicas dada por um modelo seja aquela apresentada na Tabela abaixo.

Tipo \ Região	S1	S2	S3
Irregulares	10%	25%	15%
Espirais	60%	40%	55%
Elípticas	30%	35%	30%

Em cada região será realizado um *survey* (S1, S2 e S3) considerando para cada uma um telescópio. Supondo que os telescópios são equivalentes e que as observações serão aleatórias calcule as seguintes probabilidades:

i) A primeira galáxia observada ser espiral ou elíptica.

$$P(Irreg) = \frac{10 + 25 + 15}{300} * 100 = 16{,}67\%$$

$$P(Esp\ ou\ Elip) = 100\% - 16{,}67\% = 83{,}33\%$$

ii) Se a primeira galáxia observada for irregular, qual a probabilidade dela pertencer à região do *survey* S1.

$$P(S1\ \cup\ I) = P(S1)\ + P(I) = 16{,}66 + \left(\frac{10}{300} * 100\right)$$

$$P(S1\ \cup\ I) = 19{,}99\ \%$$

3.2. Considere o exercício anterior e crie um "*bootstrap*" para gerar 10 amostras contendo 200 galáxias cada uma. Considere os valores de morfologia caracterizados pelo parâmetro g1 (da técnica *gradiente pattern analysis*) dado na tabela abaixo.

Irregulares: 1,97 – 1,99

Espirais: 1,96 – 1,98

Elípticas: 1,92 – 1,96

Aplique o Teorema de Bayes para encontrar a máxima verossimilhança considerando os modelos Gaussiano.

Algoritmo gerado (Bootstrap.py): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%203/Bootstrap.py

4. Teorema do Valor Extremo

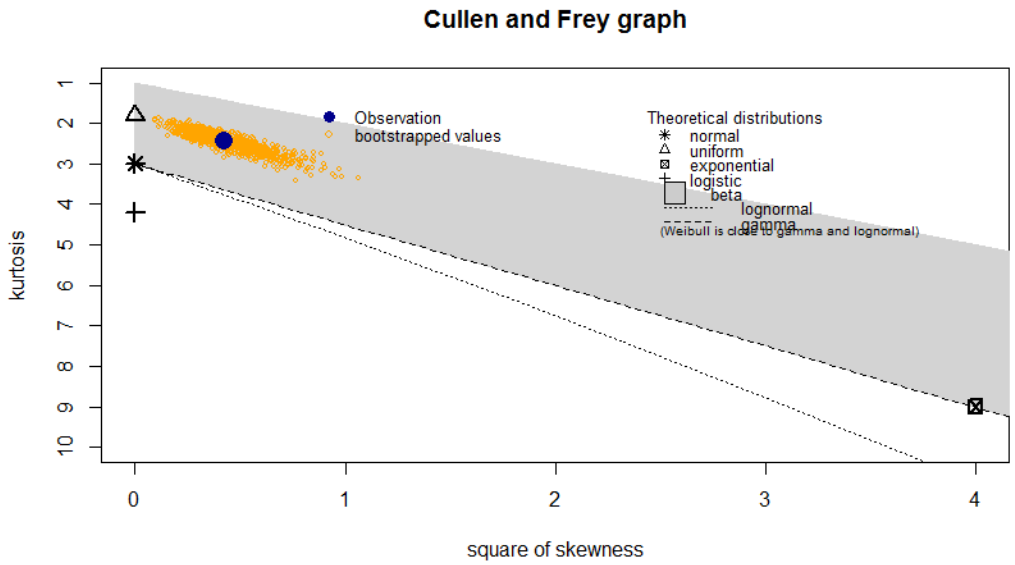
Reconsidere o exercício anterior e aplique o Teorema de Bayes para encontrar a máxima verossimilhança considerando o modelo GEV.

5. Classificação de Cullen-Frey

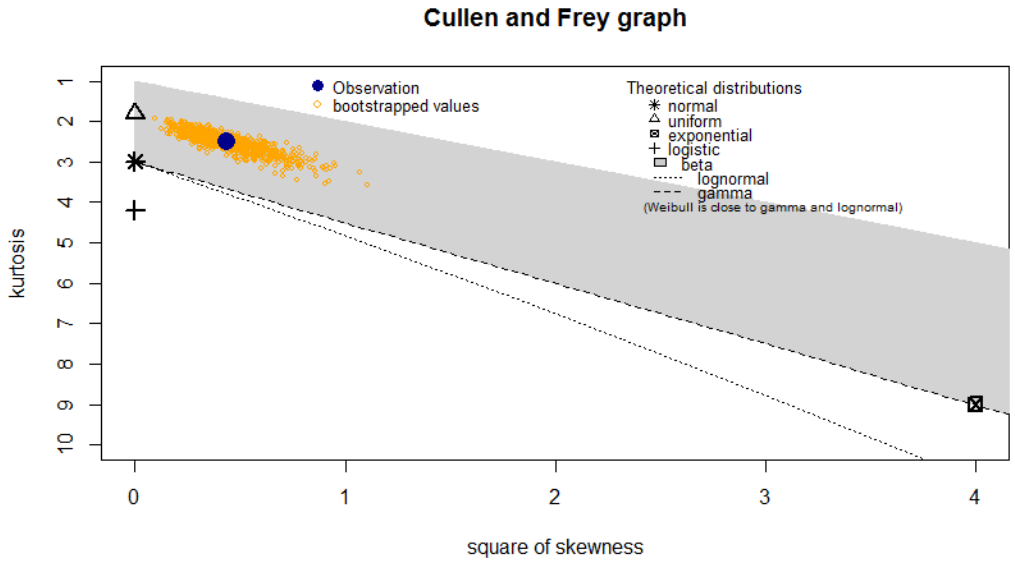
Classifique a população de amostras geradas no exercício 3.2 no espaço de Cullen-Frey, calcule os desvios e compare os desempenhos dos modelos.

Amostra	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Desvio Padrão	0.019	0.018	0.018	0.019	0.017	0.018	0.018	0.019	0.018	0.017
Skewness	-0.649	-0.661	-0.806	-0.715	-0.809	-0.810	-0.722	-0.690	-0.742	-0.638
Kurtosis	2.428	2.508	2.698	2.391	2.899	2.761	2.669	2.510	2.563	2.499

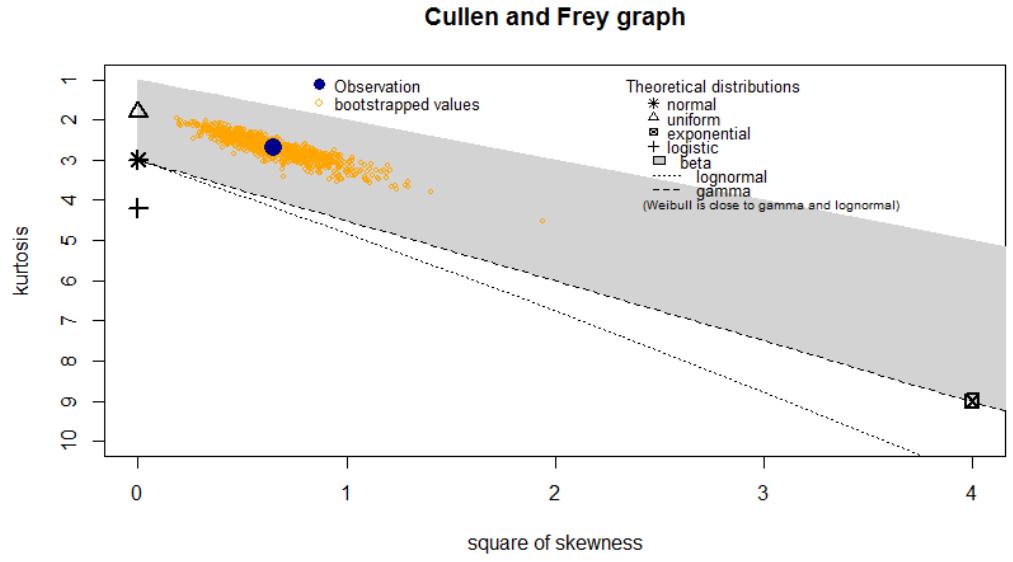
AMOSTRA: 1



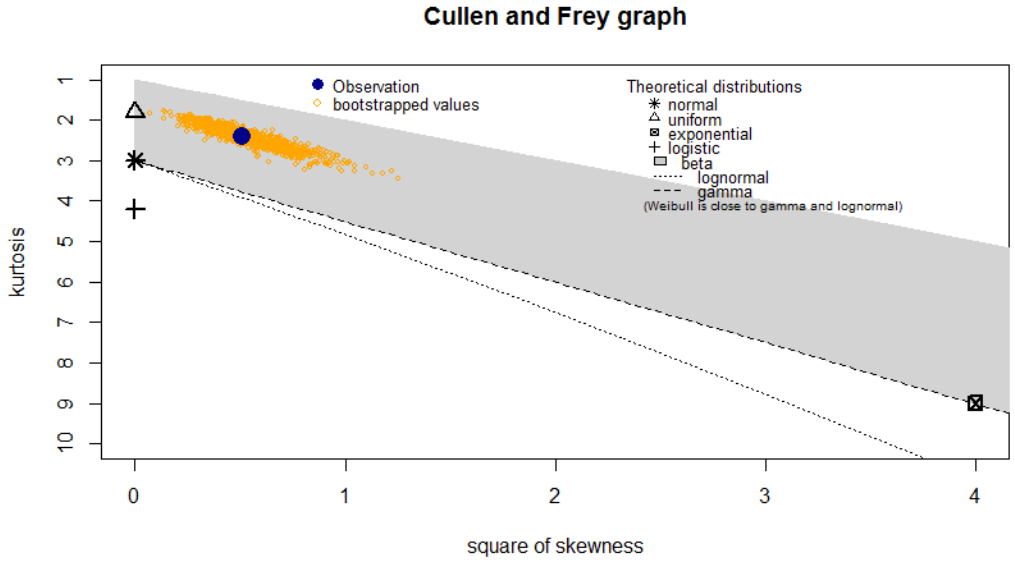
AMOSTRA: 2



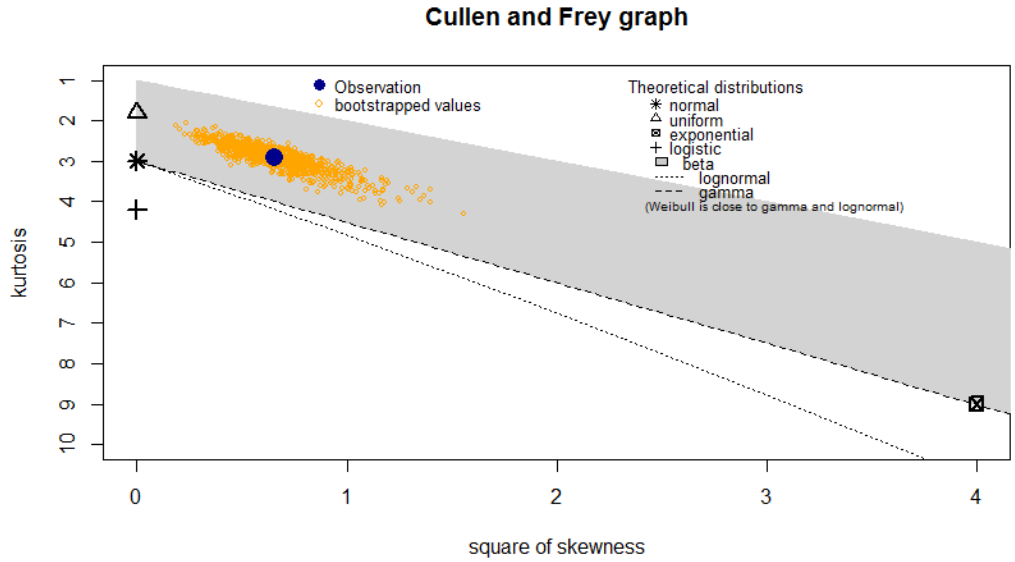
AMOSTRA: 3



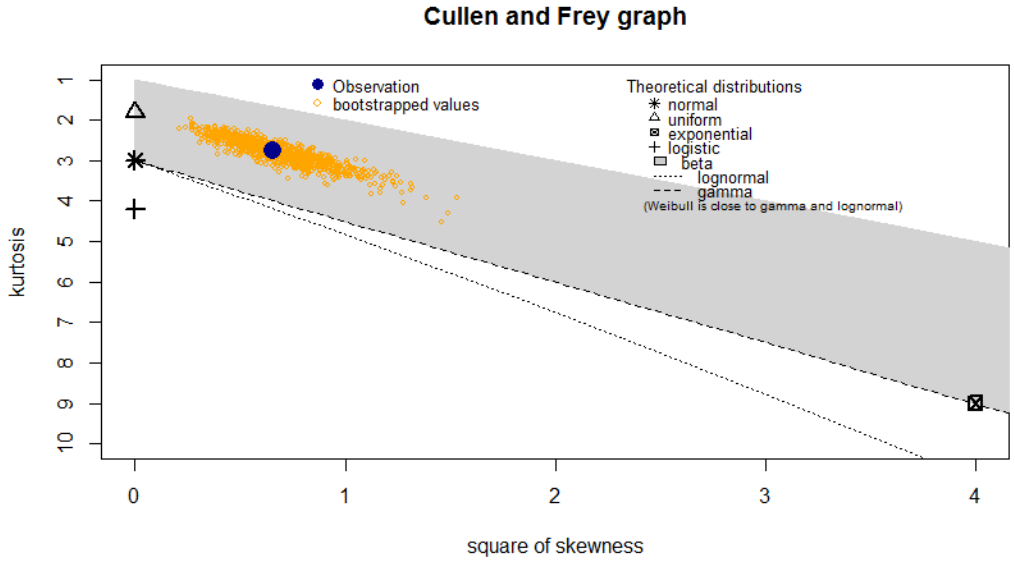
AMOSTRA: 4



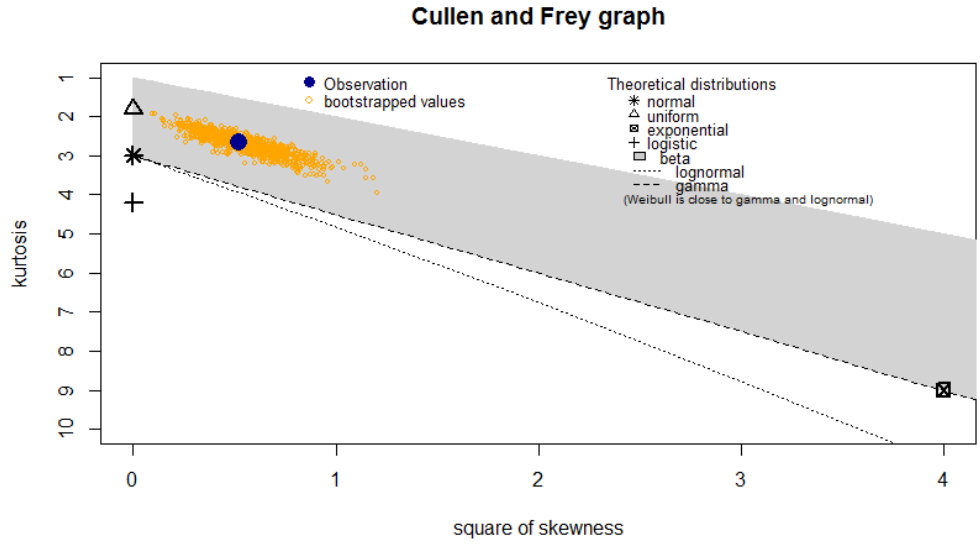
AMOSTRA: 5



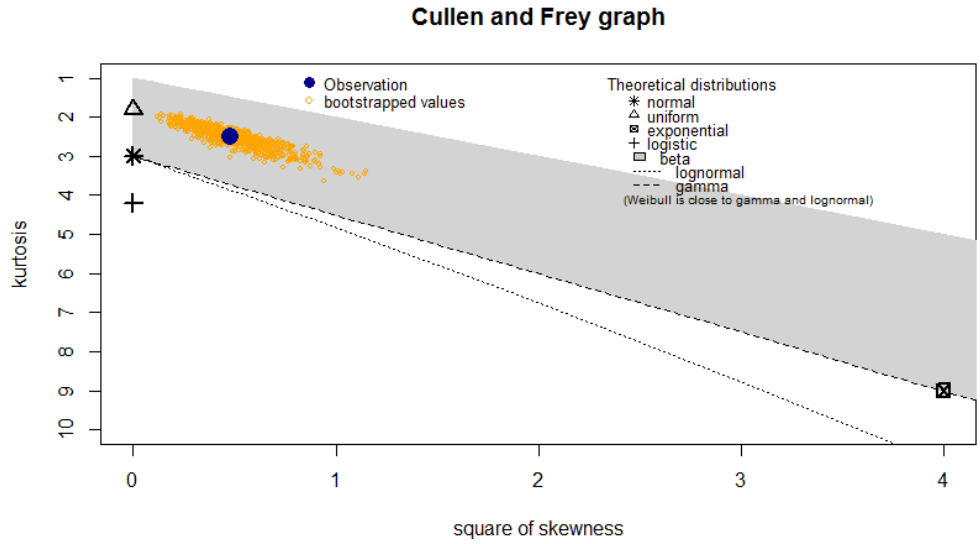
AMOSTRA: 6



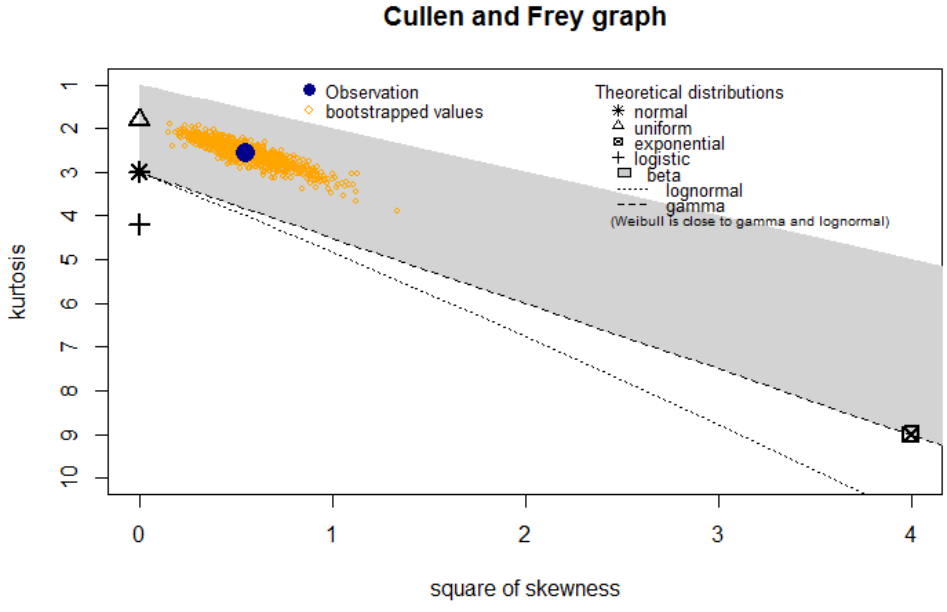
AMOSTRA: 7



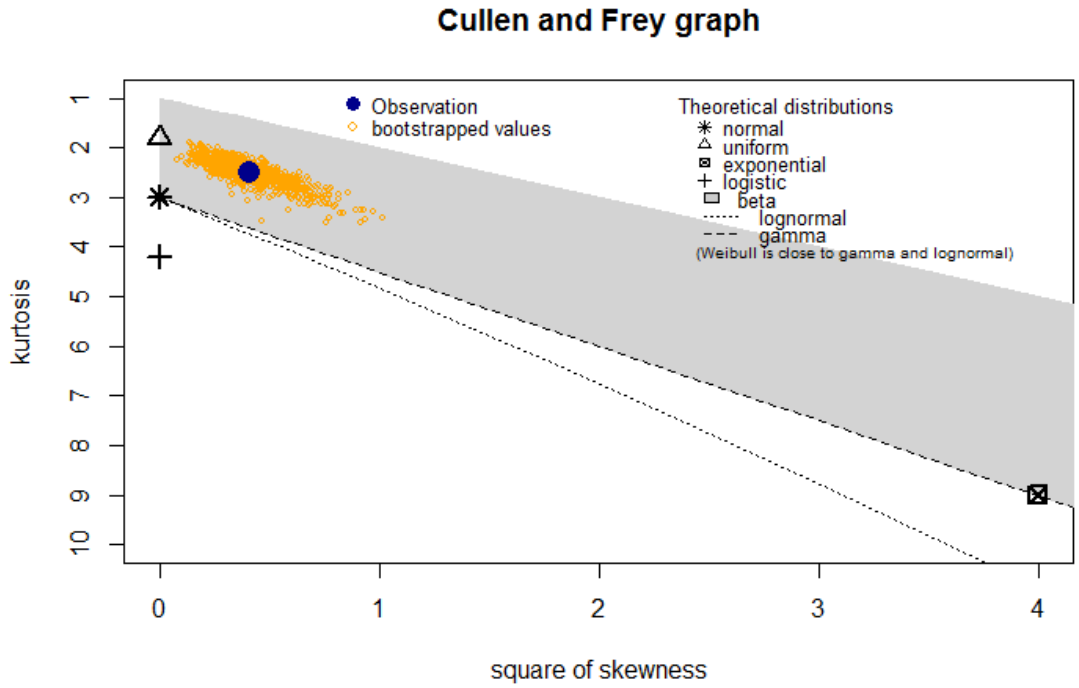
AMOSTRA: 8



AMOSTRA: 9



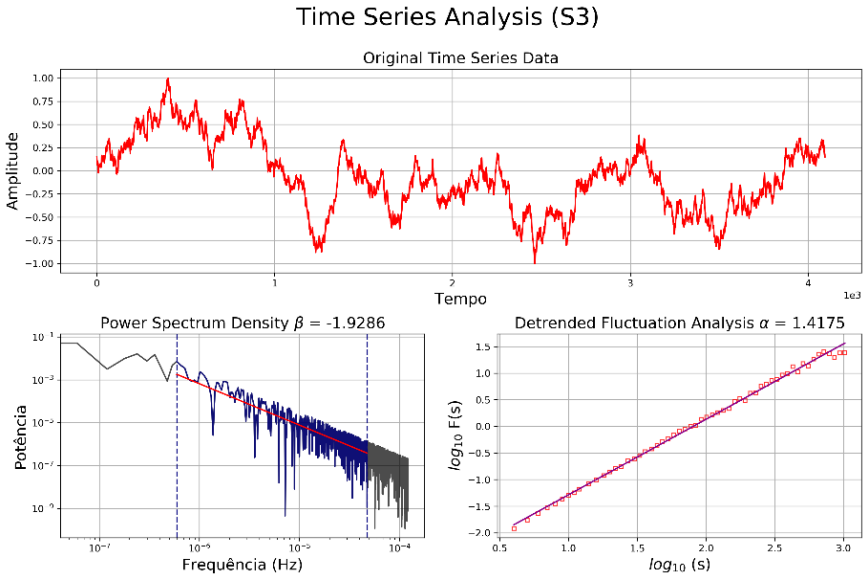
AMOSTRA: 10

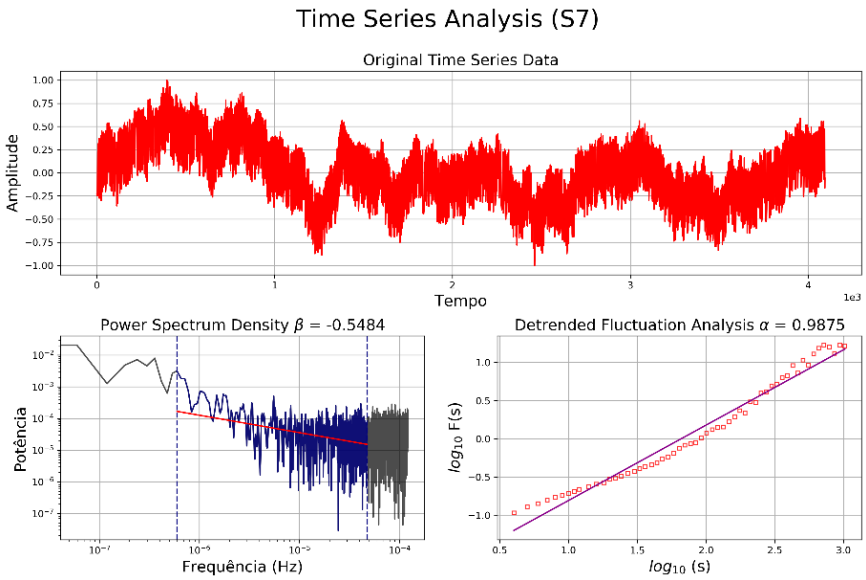


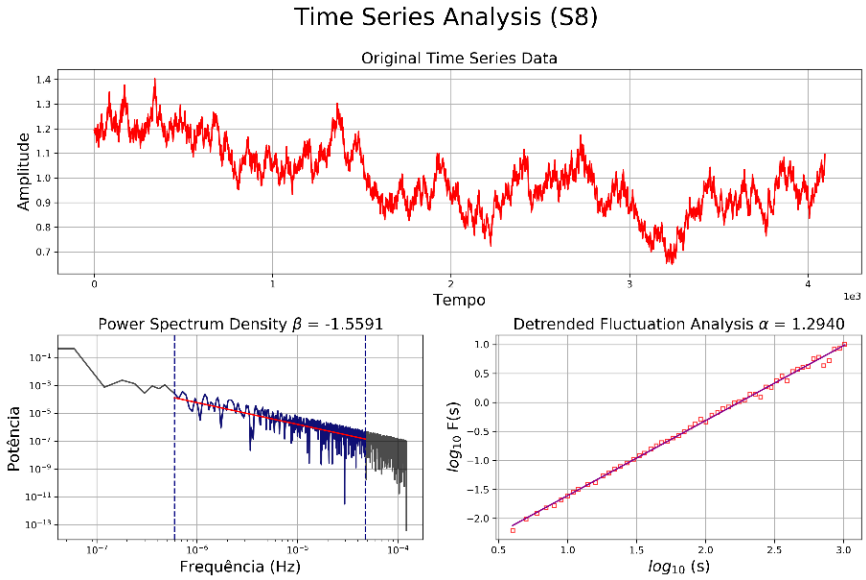
6. PSD & DFA: S3, S7, S8

6.1. Considere as séries temporais listadas acima e obtenha os valores dos respectivos índices espectrais: β (via PSD) e α (via DFA).

Algoritmo *Plota_Versao_2_SERIE_PSD_DFA.py* (Python): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%206/Plota_Versao_2_SERIE_PSD_DFA.py





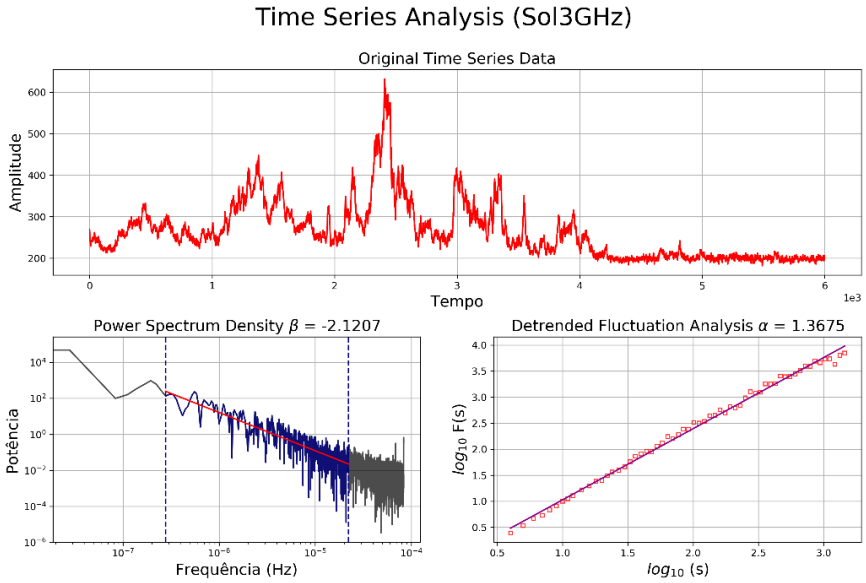


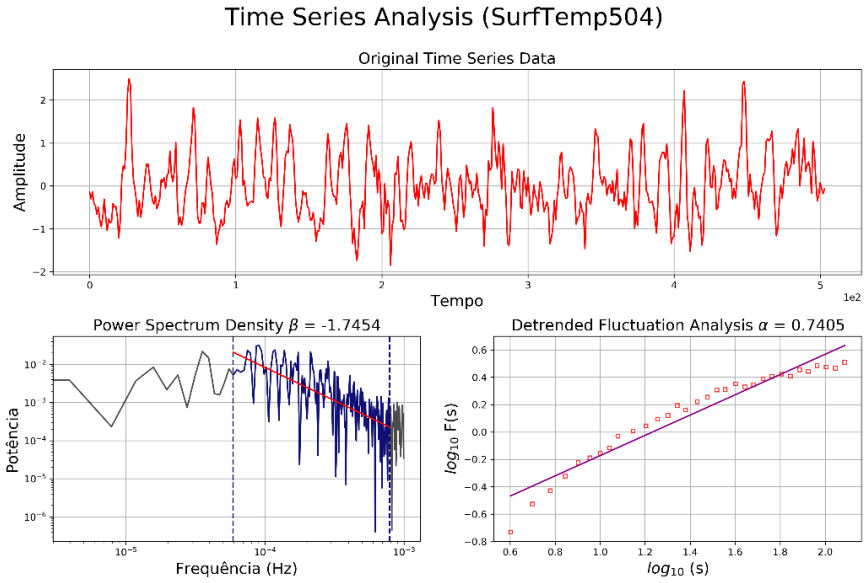
6.2. Confira se o PSD está bem ajustado a partir da fórmula WKP: β = 2 α – 1.

S3: $\beta_{S3} = 2 * 1{,}4175 - 1 \quad \beta_{S3} = 1{,}835$
S7: $\beta_{S7} = 2 * 0{,}9875 - 1 \quad \beta_{S7} = 0{,}975$
S8: $\beta_{S8} = 2 * 1{,}2940 - 1 \quad \beta_{S8} = 1{,}588$

A partir da fórmula WKP é possível observar que o PSD está bem ajustado para as séries S3 e S8. Os valores obtidos para o parâmetro β, a partir da equação WKP, se aproximam dos valores obtidos pelo PSD. Neste caso, considera-se como aceitável uma variação de +/-0,2 em relação aos valores de referência para cada série. O mesmo não ocorre para a série S7. Portanto, pode-se dizer que o PSD não está bem ajustado para a série S7.

6.3. Repita 6.1. para (a) ST-Sol3GHz, (b) ST-surftemp504 e (c) uma ST de sua escolha (alternativa)





7. Singularity Multifractal Spectra (SMS), também conhecido como MDFA.

7.1. Considere o programa mfdfa.py. Aprimore o programa para que o mesmo calcule o índice $\Psi = \Delta\alpha/\alpha_{max}$

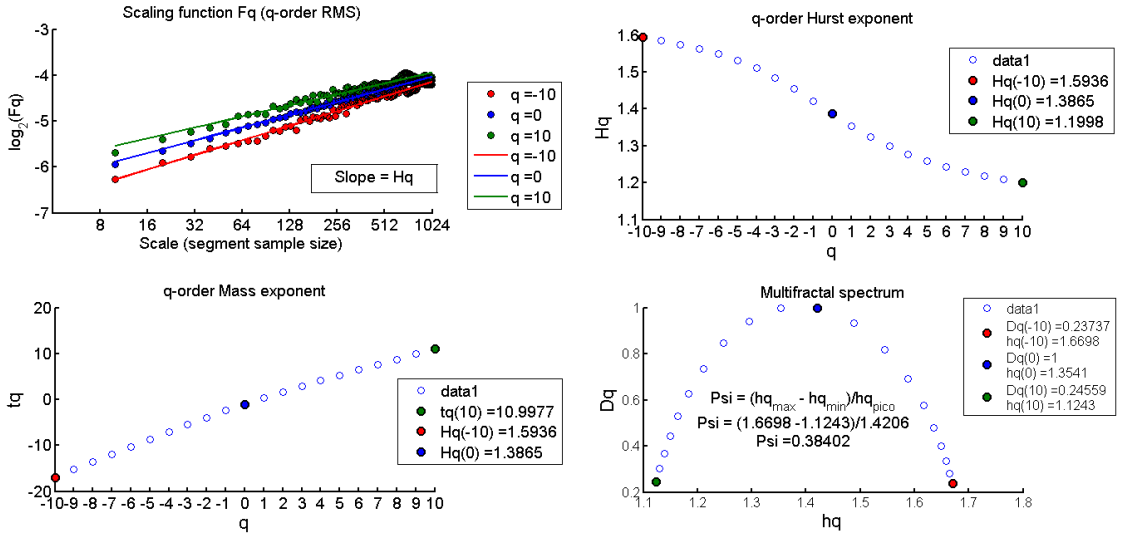
Devido a algumas dificuldades encontradas ao utilizar o programa mfdfa.py, este exercício foi resolvido utilizando o algoritmo disponibilizado pela Neela, em

Matlab. Este algoritmo foi alterado para cálculo do índice solicitado. Os resultados obtidos para as séries S3, S7 e S8 são apresentados no próximo exercício.

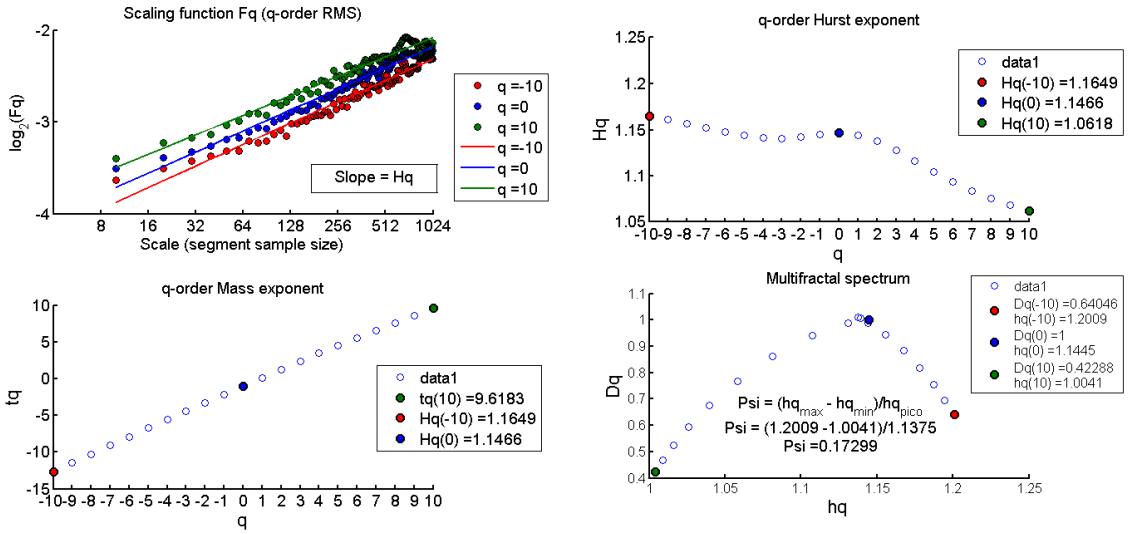
Este algoritmo pode ser encontrado no seguinte endereço: <https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%207/MFDFA4/mdfa_main.m>.

7.2. Obtenha o espectro de singularidade para todos os sinais do exercício 6.

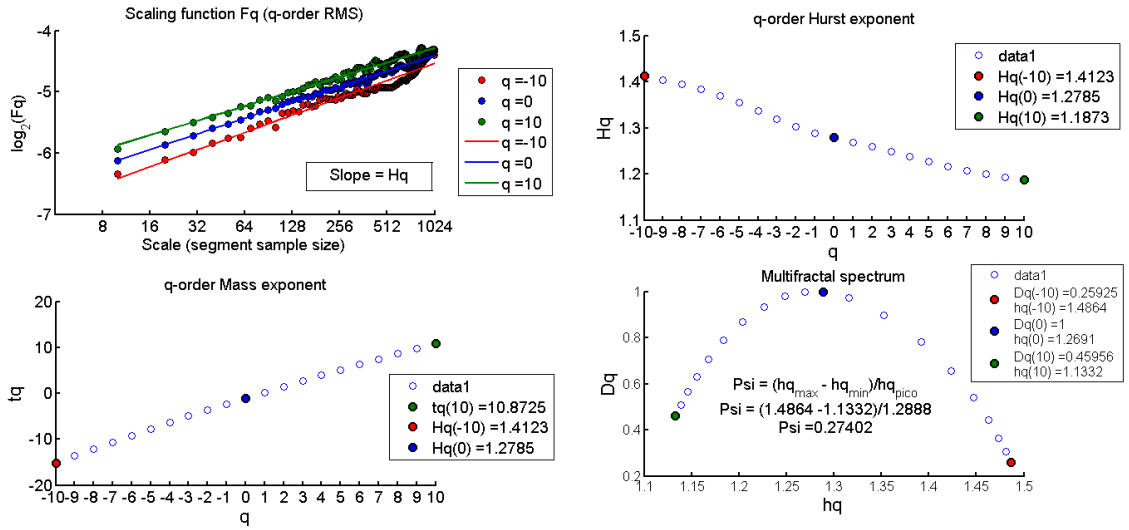
S3



S7



S8



7.3. Com base nos valores obtidos em 7.2, discuta os possíveis processos subjacentes para cada ST.

Os espectros de singularidade plotados para as séries S3, S7 e S8 indicam que estas séries são multifractais. O espectro de singularidade para S3 resultou em um espectro multifractal por se tratar de um ruído vermelho (*browniano)*, onde o coeficiente $\beta = 2$. O espectro de singularidade para S7 resultou em um espectro multifractal pois S7 é resultante da soma entre as séries S3 e S4 (ruído vermelho e série caótica, respectivamente). A série S8 corresponde a uma turbulência homogênia e, por isso, também se trata de um sinal multifractal.

8. Considere o software SPECTRUM e discuta, de forma comparativa e sucinta, outros métodos para análise espectral de sinais não abordados no curso.

Spectrum é uma biblioteca desenvolvida em Python que contém ferramentas para estimar a Densidade Espectral de Potência (PSD, *Power Spectral Density*), a partir de métodos baseados na Transformada de Fourier, multijanelamento (do inglês, *Multitapering*), métodos paramétricos ou a partir de métodos baseados em análise de autovalores. Informações sobre o Spectrum podem ser encontradas no seguinte endereço: <https://pyspectrum.readthedocs.io>.

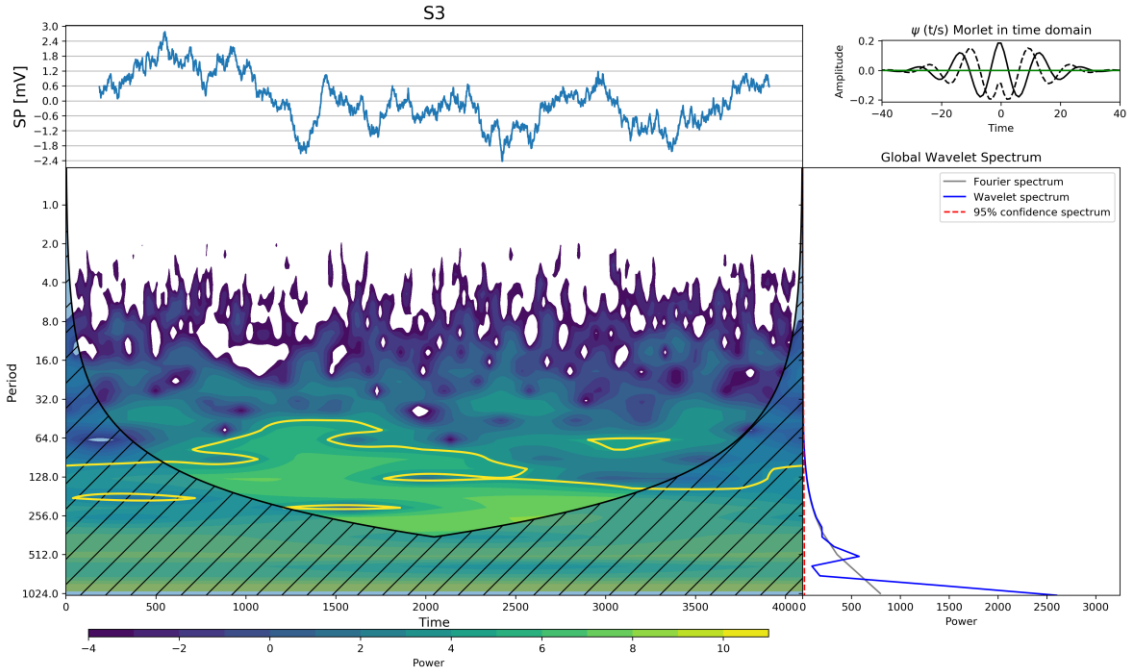
Os métodos de Fourier são baseados em correlogramas, periodogramas e estimadores de Welch. Métodos padrões de janelamento (do inglês, *tapering*) estão

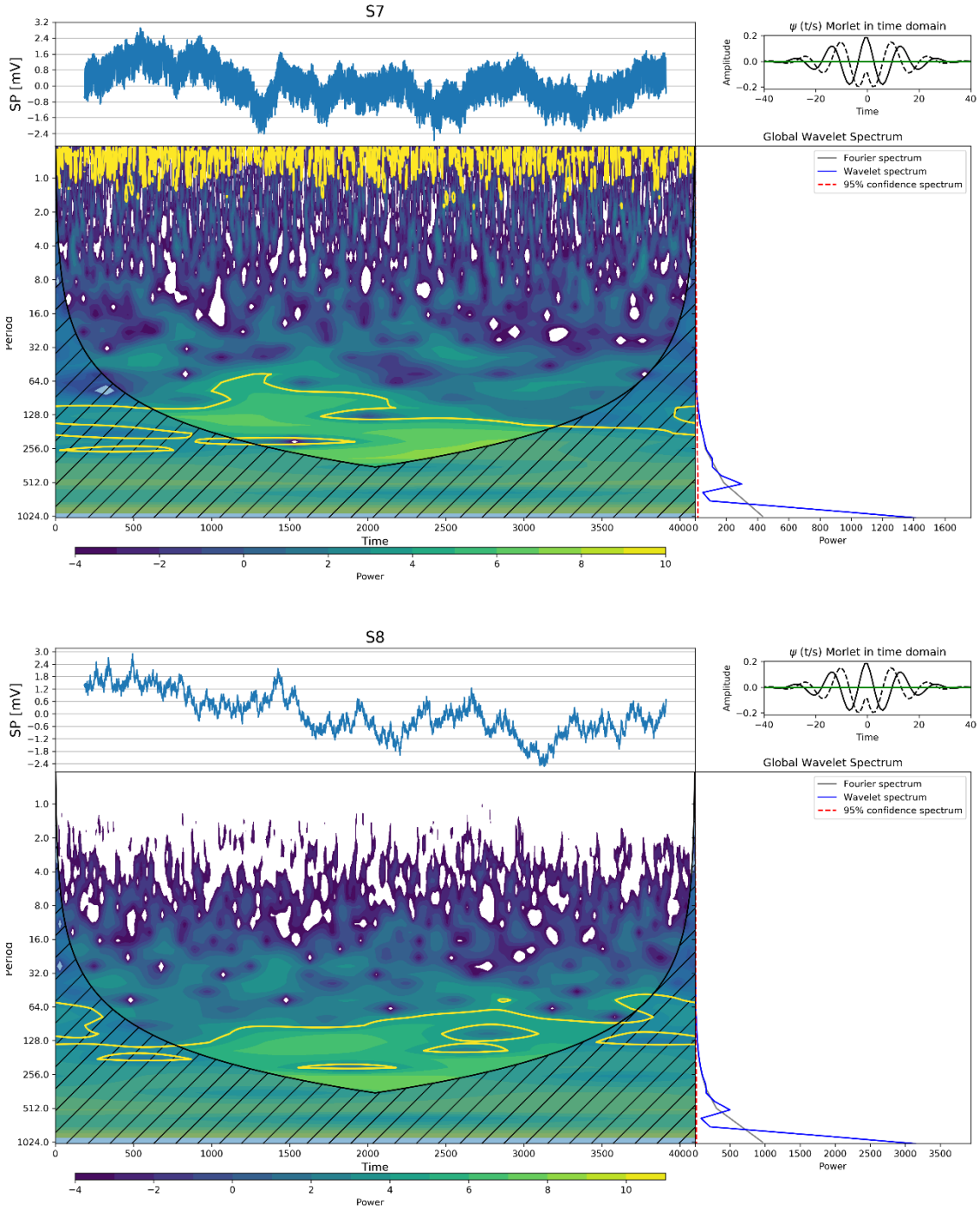
também disponíveis, tais como *Hann*, *Hamming* e *Blackman*). Podem também ser utilizados métodos não paramétricos baseados em análise de autovalores (e.g., MUSIC), análise de variância mínima (e.g., Capon) e entropia máxima (MEM). O software Spectrum inclui também três métodos paramétricos, baseados em *Yule-Walker*, BURG e métodos de covariância, os quais permitem estimar parâmetros de autorregressivos (AR). Os métodos Yule-Walker e BURG permitem minimizar erros de predição por meio da recursão de Levinson. O método de multijanelamento (do inglês, *Multitapering*) também pode ser utilizado para estimativa espectral. Este método permite preservar uma boa resolução, enquanto reduz tanto o viés quanto a variância.

9. Global Wavelet Spectrum

9.1. Utilize o Waipy para obter o GWS (Morlet) de todos as ST do exercício 6.1.

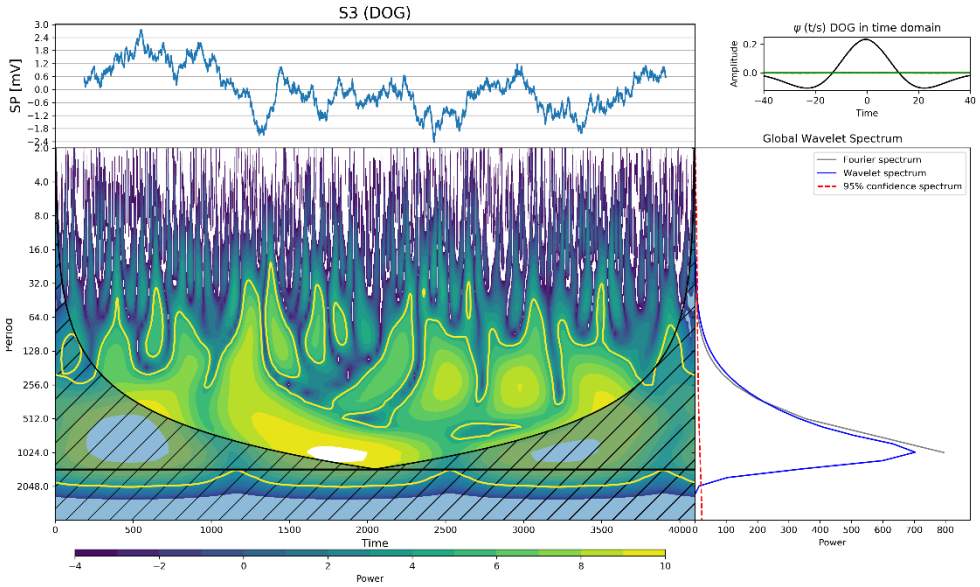
Algoritmo Morlet_Waipy.py (Python): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%209/Morlet_Waipy.py

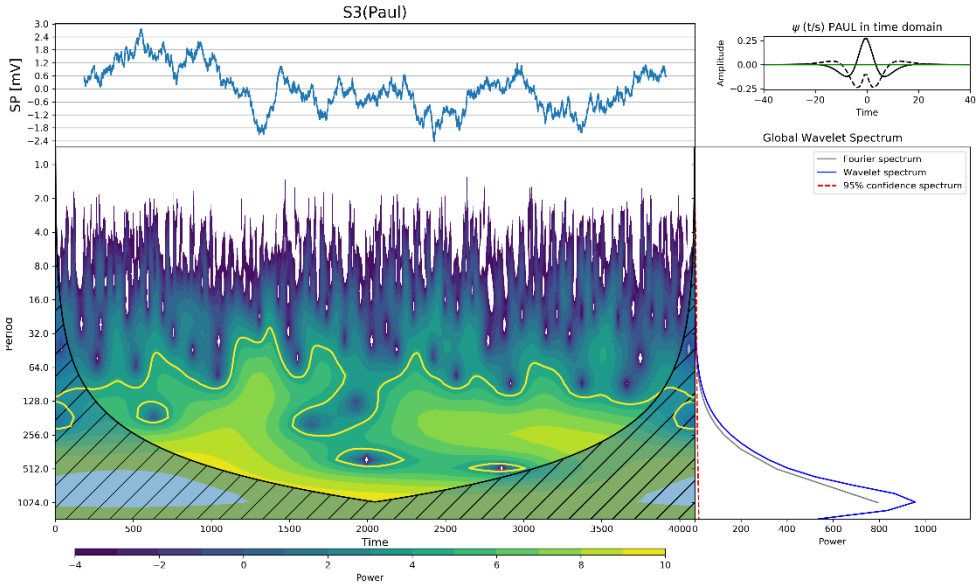




9.2. Repita 9.1 utilizando uma Db8.

O Waipy permite obter o GWS utilizando apenas as seguintes transformadas contínuas. Portanto, não é possível utilizar uma Db8 (transformada discreta) no Waipy. Este pacote permite utilizar as seguintes transformadas: Morlet, DOG (do inglês, *Derivative of Gaussian*) e Paul. As figuras abaixo apresentam os resultados obtidos ao se aplicar as transformadas DOG e Paul sobre a série S3.





10. Self-Organized Criticality (SOC)

i) Calcule a Taxa Local de Flutuação [Yi] para cada valor da ST

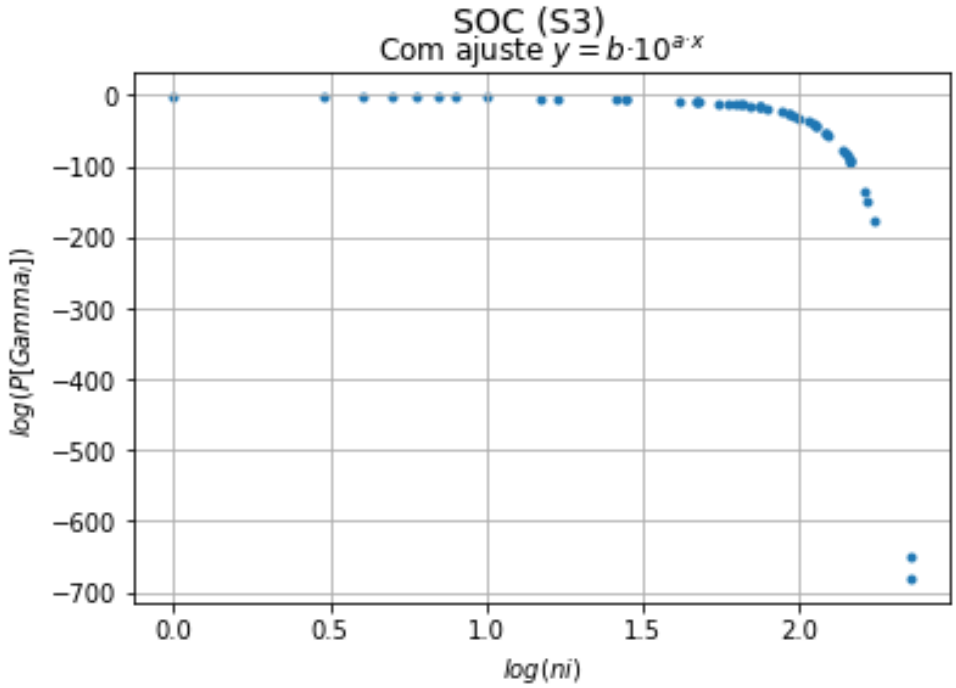
ii) Calcule $P[Yi] = counts(ni) / N$

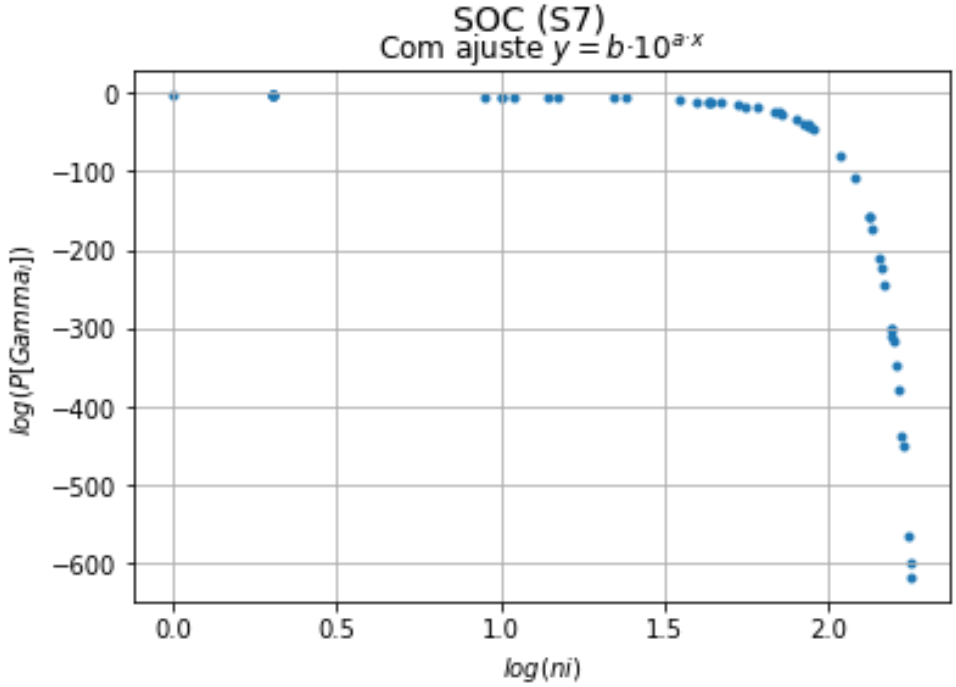
iii) Plot $logP[Yi]$ x $log\ ni$ (e ajuste uma lei de potência)

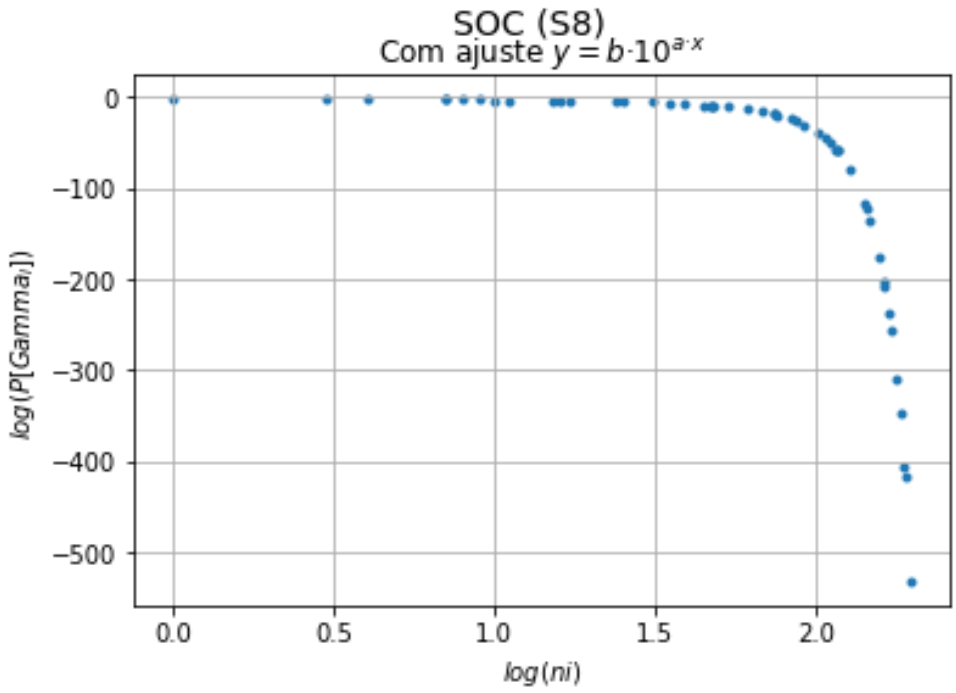
10.1. Implemente um algoritmo em Python para caracterização de SOC a partir de uma ST.

Algoritmo SOC.py (Python): https://github.com/wivoliveira/CAP-239/blob/master/Exercicio%2010/SOC.py

10.2. Aplique o SOC.py para todas as ST do exercício 6.1.







REFERÊNCIAS

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning.** Ed. 2. New York: Springer Series in Statistics Springer New York Inc. 739 p., 2009.

IHLEN, E. A. Introduction to multifractal detrended fluctuation analysis in Matlab. **Frontiers in Physiology,** v. 3, n. 141, 2012. doi: 10.3389/fphys.2012.00141

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning with applications in R.** Springer: New York, 2013.

TORRENCE, C.; COMPO, G. P. A practical guide to wavelet analysis. **Bulletin of the American Meteorological Society,** v. 79, n. 1, p. 61-78, 1998.