Lista 1: Um pouco mais sobre CDFs...

Forma de entrega

Salve essa página (com código e respostas geradas) em `.pdf` e submeta o documento na atividade do AVA (apenas um integrante do grupo deve fazer a submissão).

Grupos

- **Graduação:** grupo de até 3 pessoas
- **Pós-Graduação:** individual

Integrantes do grupo

1. **Nome(s):** Alexandre Rosseto Lemos
2. **Matrícula(s):** 2021231580
3. **Curso(s):** PPGI (Mestrado)
4. **Link do seu Colab com suas soluções (para que o professor possa acessar e rodar o código):** https://colab.research.google.com/drive/15SO_B6mRAAQuKx2O5objuu-pT647U9Jo?usp=sharing

Questão 1

O objetivo desta questão é comparar as **Funções de Distribuição** (i.e., CDF) teórica e empírica de algumas distribuições contínuas bem conhecidas.

Lembre-se que nós conversamos um pouco sobre CDFs na Aula 3.

Exemplo

Vamos começar com um exemplo sobre a distribuição Uniforme contínua.

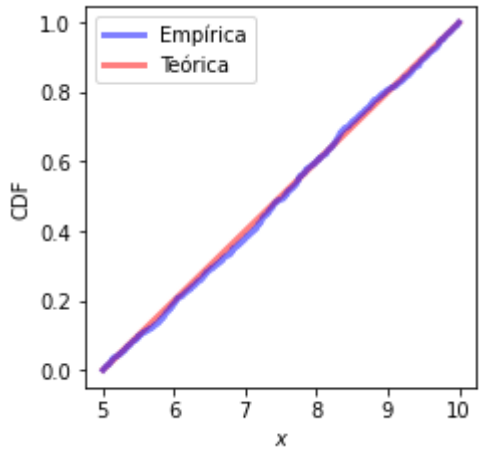
A biblioteca `scipy` possui funcionalidades para gerar números pseudo-aleatórios e para computar valores da CDF teórica da distribuição. Leia mais sobre em https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.uniform.html.

A biblioteca `statsmodels` possui funcionalidades para computar a CDF empírica de um conjunto de observações

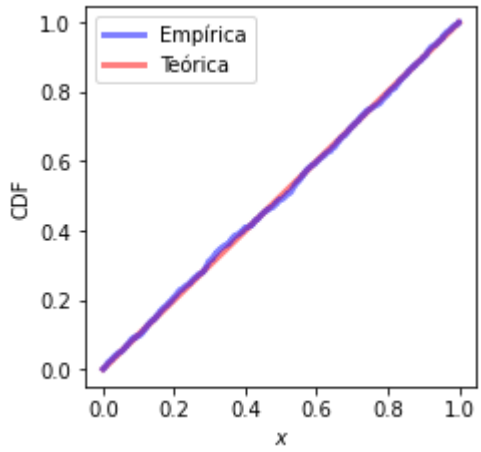
```
from scipy.stats import uniform
import matplotlib.pyplot as plt
from statsmodels.distributions.empirical_distribution import ECDF
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarni
  import pandas.util.testing as tm
```

```
def plot_uniform_cdf(loc, scale, n):
  # gera 'n' números de uma distribuição uniforme contínua no
  # intervalo [loc, loc + scale]
  r = uniform.rvs(loc = loc, scale = scale, size = n)

  # plota a CDF empírica dos números gerados
  cdf = ECDF(r)
  plt.plot(cdf.x, cdf.y, linewidth = 3, alpha = 0.5, color = 'blue',
           label = 'Empírica')

  # plota a CDF teórica de uma distribuição contínua uniforme no intervalo
  # [loc, loc + scale]
  plt.plot(cdf.x, uniform.cdf(cdf.x, loc = loc, scale = scale),
           alpha = 0.5, label = 'Teórica', color = 'red',
           linewidth = 3, zorder = 0);

  # Ajustes de rótulos e tamanho da figura...
  plt.legend(loc = 'best');
  plt.xlabel(r'$x$');
  plt.ylabel(r'CDF');
  plt.gcf().set_size_inches(3.5, 3.5)


plot_uniform_cdf(5, 5, 1000)
```



```
plot_uniform_cdf(0, 1, 1000)
```



Parte (*a*)

Repita a análise acima para a distribuição normal. Lembre-se que a distribuição normal possui dois parâmetros: μ e σ.

Consulte https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html para gerar números pseudo-aleatórios e para computar a CDF da distribuição normal.

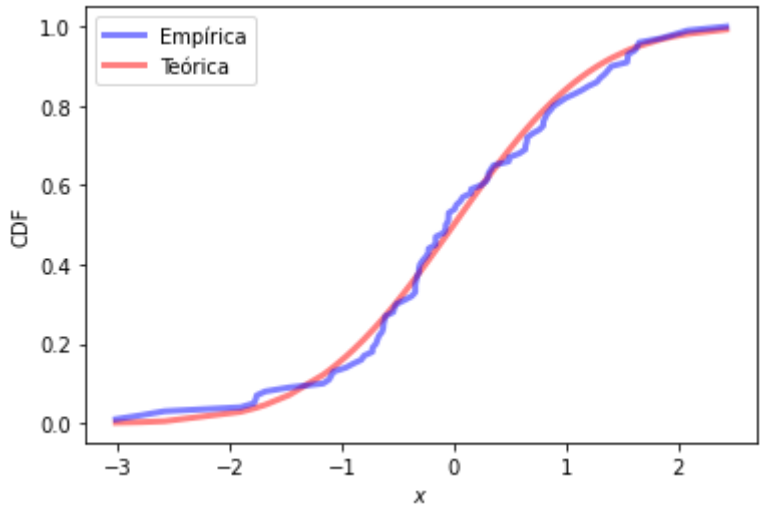
Você deve:

1. Completar a função abaixo para gerar gráficos comparativos entre as CDFs empírica e teórica da distribuição normal;
2. Executar seu código, variando os parâmetros da distribuição e número de elementos gerados.

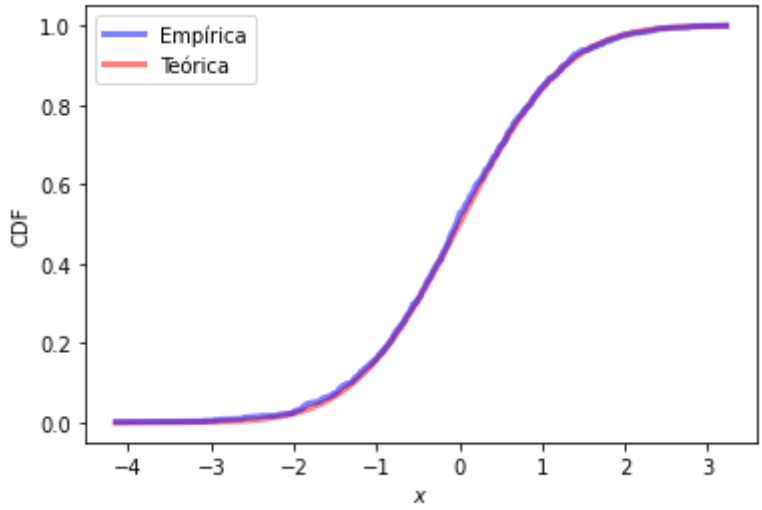
```
from scipy.stats import norm


def plot_normal_cdf(mu, sigma, n):
  '''
  Info:
    Essa funcao gera n valores pseudo-aleatorios de uma distribuicao normal
    com media = mu e desvio padrao = sigma e plota a comparacao entre as
    CDFs empírica e teórica da distribuição normal.
  ----------
  Input:
    mu: média da gaussiana
    sigma: desvio padrão da gaussiana
    n: quantidade de amostras geradas
  ----------
  Output:
    None
  '''
  # Gerando as amostras pseudo-aleatorias
  r = norm.rvs(loc = mu, scale = sigma, size = n)
```

```
  # Plotando a CDF empírica dos números gerados
  cdf = ECDF(r)
  plt.plot(cdf.x, cdf.y, linewidth = 3, alpha = 0.5, color = 'blue',
           label = 'Empírica')

  # Plotando a CDF teórica de uma distribuição normal
  plt.plot(cdf.x, norm.cdf(cdf.x, loc = mu, scale = sigma),
           alpha = 0.5, label = 'Teórica', color = 'red',
           linewidth = 3, zorder = 0);

  # Ajustes de rótulos e tamanho da figura
  plt.legend(loc = 'best');
  plt.xlabel(r'$x$');
  plt.ylabel(r'CDF');
  plt.show()
```

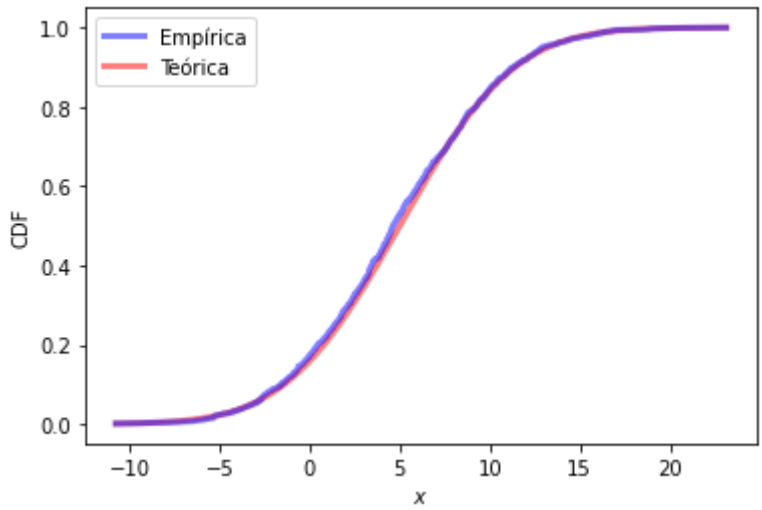
```
import matplotlib.pyplot as plt
plot_normal_cdf(0, 1, 100)
```




```
plot_normal_cdf(0, 1, 1000)
```




```
plot_normal_cdf(5, 5, 1000)
```



Parte (*b*)

Repita a análise inicial para a distribuição exponencial. Lembre-se, dado um parâmetro $\lambda > 0$, a densidade da distribuição exponencial é:

$$f(x) = \lambda e^{-\lambda x}, x > 0.$$

Consulte https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.expon.html para gerar números pseudo-aleatórios e para computar a CDF da distribuição exponencial.

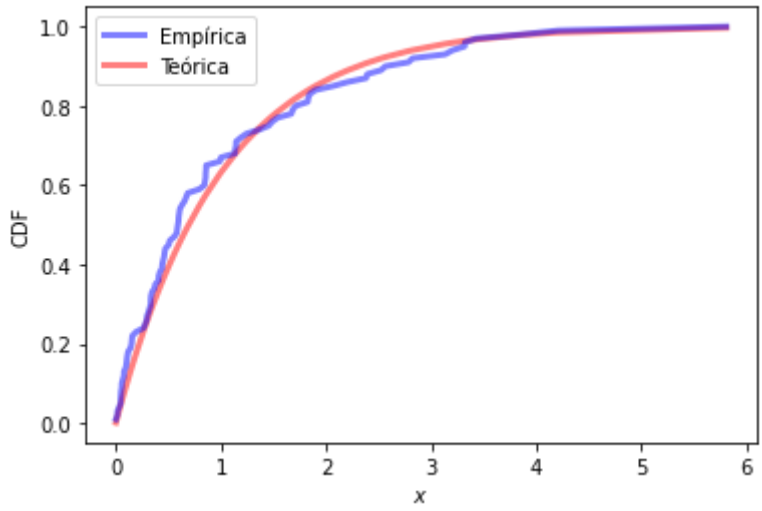
Você deve:

1. Completar a função abaixo para gerar gráficos comparativos entre as CDFs empírica e teórica da distribuição exponencial;
2. Executar seu código, variando os parâmetros da distribuição e número de elementos gerados.

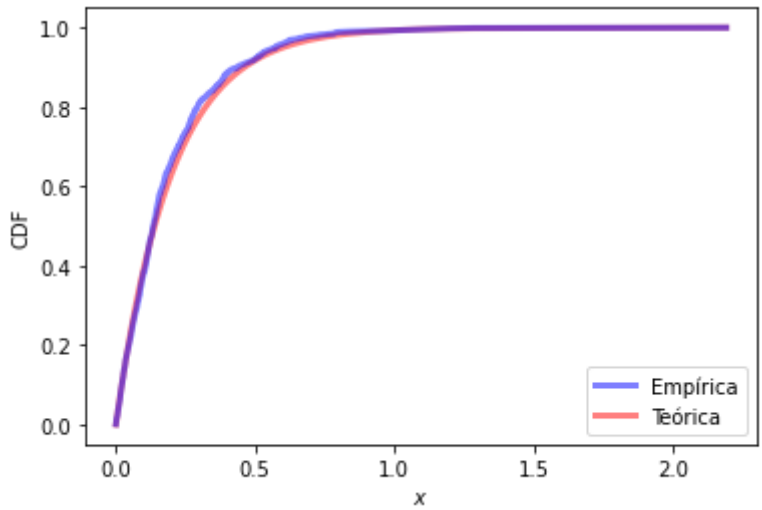
```
from scipy.stats import expon


def plot_exponential_cdf(lambda_, n):
  '''
  Info:
    Essa funcao gera n valores pseudo-aleatorios de uma distribuicao exponencial
    com o parametro de taxa lambda = lambda_ e plota a comparacao entre as
    CDFs empírica e teórica da distribuição exponencial.
  ----------
  Input:
    lambda_: parametro de taxa
    n: quantidade de amostras geradas
  ----------
  Output:
    None
  '''
  # Gerando as amostras pseudo-aleatorias
  r = expon.rvs(scale = (1/lambda_), size = n)
```

```
  # Plotando a CDF empírica dos números gerados
  cdf = ECDF(r)
  plt.plot(cdf.x, cdf.y, linewidth = 3, alpha = 0.5, color = 'blue',
           label = 'Empírica')

  # Plotando a CDF teórica de uma distribuição exponencial
  plt.plot(cdf.x, expon.cdf(cdf.x, scale = (1/lambda_)),
           alpha = 0.5, label = 'Teórica', color = 'red',
           linewidth = 3, zorder = 0);

  # Ajustes de rótulos e tamanho da figura
  plt.legend(loc = 'best');
  plt.xlabel(r'$x$');
  plt.ylabel(r'CDF');
  plt.show()
```

```
plot_exponential_cdf(1, 100)
```




```
plot_exponential_cdf(5, 1000)
```



Parte (*c*)

Agora as coisas ficam mais interessantes...

Repita a análise inicial para a distribuição de pareto. Lembre-se, dado um parâmetro $b > 0$, a densidade da distribuição de pareto é:

$$f(x) = \frac{b}{x^{b+1}}, x \geq 1.$$

Consulte https://docs.scipy.org/doc/scipy/tutorial/stats/continuous_pareto.html para gerar números pseudo-aleatórios e para computar a CDF da distribuição de pareto.

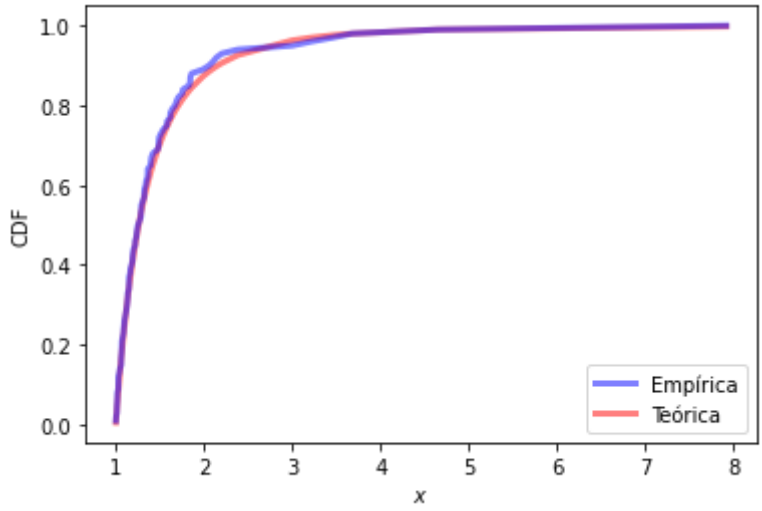
Você deve:

1. Completar a função abaixo para gerar gráficos comparativos entre as CDFs empírica e teórica da distribuição de pareto;
2. Executar seu código, variando os parâmetros da distribuição e número de elementos gerados.

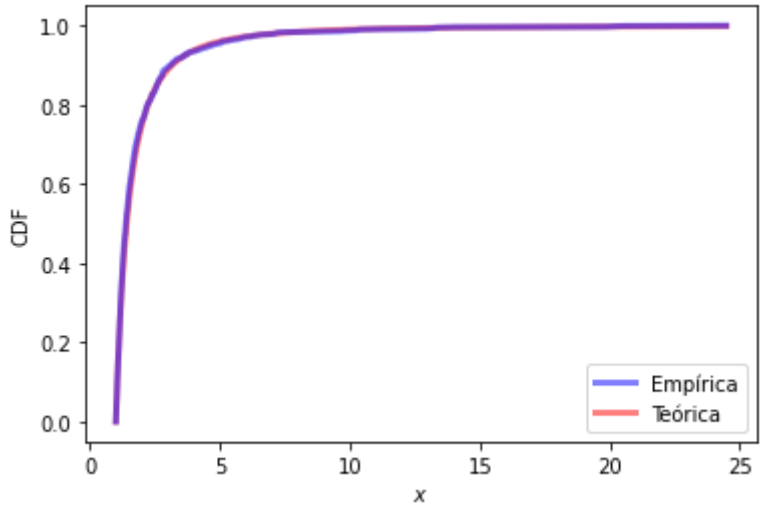
```
from scipy.stats import pareto


def plot_pareto_cdf(b, n):
  '''
  Info:
    Essa funcao gera n valores pseudo-aleatorios de uma distribuicao pareto
    com parametro de forma = b e plota a comparacao entre as
    CDFs empírica e teórica da distribuicao pareto.
  ----------
  Input:
    b: parametro de forma
    n: quantidade de amostras geradas
  ----------
  Output:
    None
  '''

  # Gerando as amostras pseudo-aleatorias
  r = pareto.rvs(b, size = n)

  # Plotando a CDF empírica dos números gerados
  cdf = ECDF(r)
  plt.plot(cdf.x, cdf.y, linewidth = 3, alpha = 0.5, color = 'blue',
           label = 'Empírica')

  # Plotando a CDF teórica de uma distribuição pareto
  plt.plot(cdf.x, pareto.cdf(cdf.x, b),
           alpha = 0.5, label = 'Teórica', color = 'red',
           linewidth = 3, zorder = 0);
```

```
  # Ajustes de rótulos e tamanho da figura
  plt.legend(loc = 'best');
  plt.xlabel(r'$x$');
  plt.ylabel(r'CDF');
  plt.show()
```

```
plot_pareto_cdf(3, 100)
```

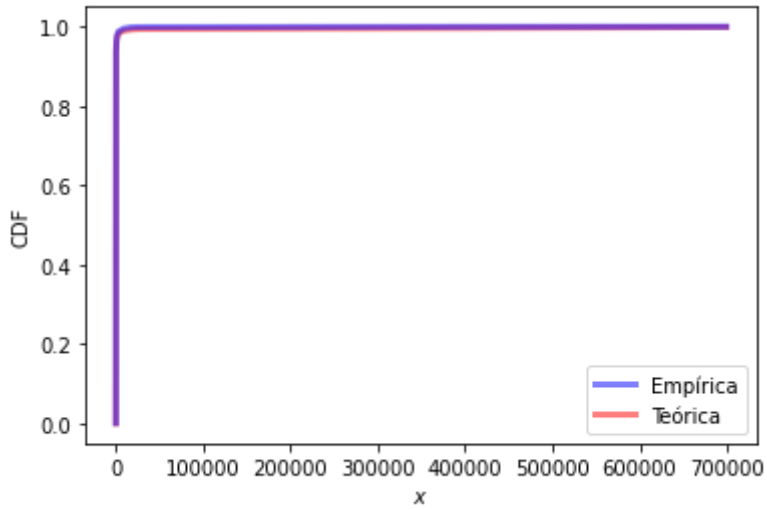



```
plot_pareto_cdf(2, 1000)
```




```
plot_pareto_cdf(1, 1000)
```


```
plot_pareto_cdf(0.5, 1000)
```



Após fazer as figuras, você provavelmente observou que para valores pequenos de b, a CDF da distribuição de pareto é bastante difícil de ler, uma vez fica praticamente "colada" às retas $x = 0$ e $y = 1$.

Esse fenômeno ocorre porque, quando b é pequeno, valores muito grandes (i.e., discrepantes ou *outliers*) têm uma probabilidade não negligível de ocorrerem.

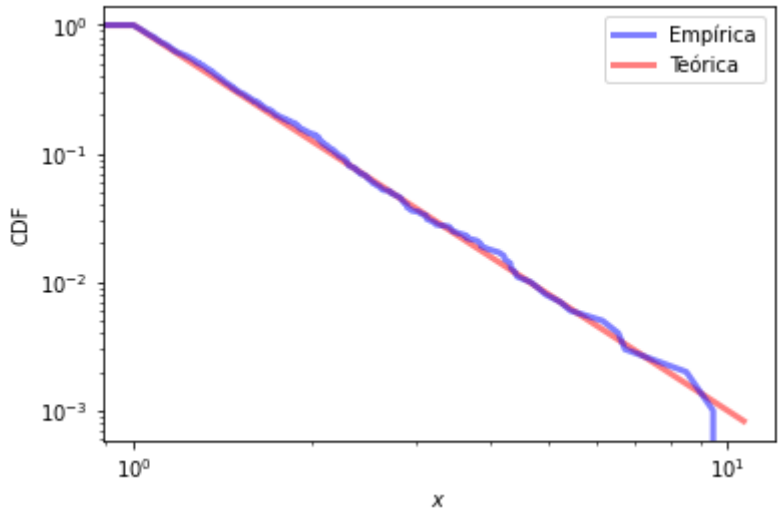
Quando observamos o comportamento das figuras acima, devemos modificar o forma de visualizar a distribuição:

1. Devemos gerar uma figura para a CCDF (*Complementary Cumulative Distribution Function*). A CCDF de um valor x é definida como 1 menos a CDF de x.
2. Devemos colocar os eixos x e y em escala logarítmica.

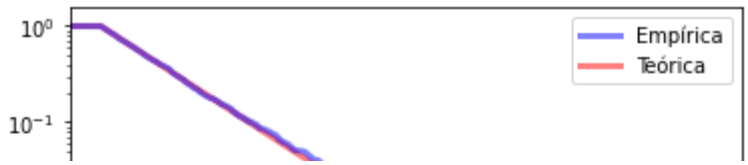
Complete a função abaixo para gerar a figura da forma descrita acima e gere os gráficos para os mesmos casos que testou anteriormente.

```
def plot_pareto_ccdf(b, n):
  '''
  Info:
    Essa funcao gera n valores pseudo-aleatorios de uma distribuicao pareto
    com parametro de forma = b e plota a comparacao entre as
    CDFs empírica e teórica da distribuicao pareto.
  ----------
  Input:
    b: parametro de forma
    n: quantidade de amostras geradas
  ----------
  Output:
    None
```

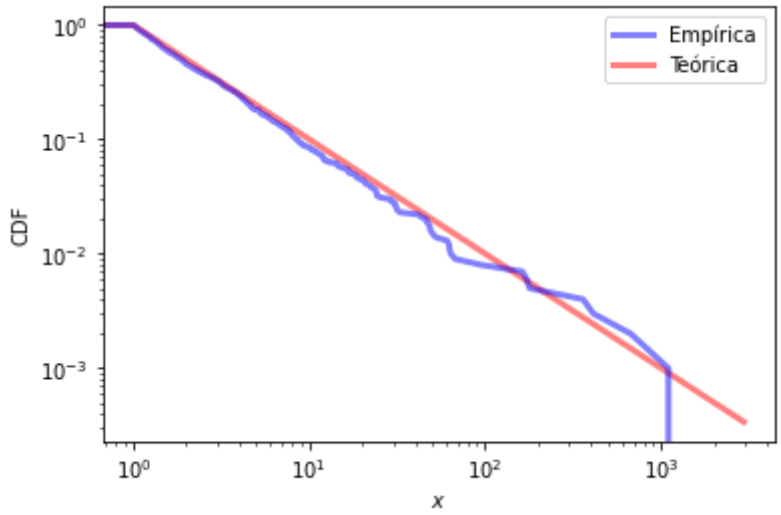
```
  '''

  # Gerando as amostras pseudo-aleatorias
  r = pareto.rvs(b, size = n)

  # Gerando a CDF empírica dos números gerados
  cdf = ECDF(r)

  plt.plot(cdf.x, (1 - cdf.y), linewidth = 3, alpha = 0.5, color = 'blue',
           label = 'Empírica')

  # Plotando a CDF teórica de uma distribuição pareto
  plt.plot(cdf.x, (1 - pareto.cdf(cdf.x, b)),
           alpha = 0.5, label = 'Teórica', color = 'red',
           linewidth = 3, zorder = 0);

  # Ajustes de rótulos e tamanho da figura
  plt.legend(loc = 'best');
  plt.xscale("log")
  plt.yscale("log")
  plt.xlabel(r'$x$');
  plt.ylabel(r'CDF');
  plt.show()


plot_pareto_ccdf(3, 1000)
```



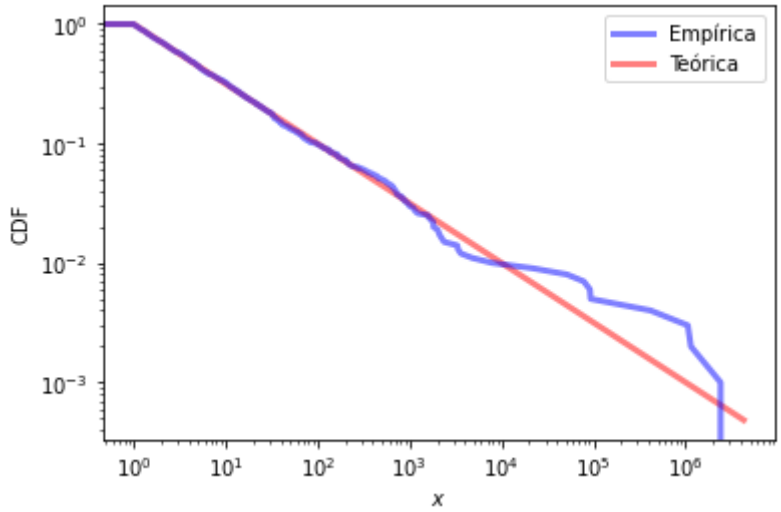

```
plot_pareto_ccdf(2, 1000)
```




```
plot_pareto_ccdf(1, 1000)
```




```
plot_pareto_ccdf(0.5, 1000)
```



Questão 2

Esta questão é **obrigatória** apenas para alunos da **pós-graduação** e, neste caso, vale 50% da nota.

O objetivo desta questão é entender o comportamento da **parte (c)** da questão anterior.

Você deve ler as duas primeiras seções do artigo https://arxiv.org/pdf/cond-mat/0412004.pdf e fazer um resumo (de no máximo uma página). Certifique-se de incluir no seu resumo, entre outras informações que achar relevantes, o motivo de a CCDF da distribuição de pareto ter a forma que tem quando ambos os eixos estão em escala logarítmica.

O assunto principal das duas primeiras seções do artigo é a distribuição de lei de potência (power law distribution), efeito que ocorre em diversos cenários na natureza, bastante diversos, desde números de pessoas vivendo em uma cidade até tamanhos de terremotos, crateras lunares e explosões solares.

Não é fácil detectar distribuição de lei de potência tanto em dados gerados por humanos quanto dados na natureza. A estratégia padrão é gerar um histograma dos dados com os eixos em escalas logarítmicas. Caso o resultado seja uma reta, os dados apresentam uma distribuição de lei de potência, cuja fórmula tem-se abaixo.

$$p(x) = Cx^{-\alpha}$$

Onde apha é chamado de expoente da lei de potência e C é uma constante.

Essa maneira, no entanto, não é a melhor para se descobrir se os dados apresentam essa distribuição, pois ao transformar as escalas dos eixos para logarítmicas, podem surgir áreas na curva que não são uma reta por conta das distorções causadas por erros e ruídos nas amostras, especialmente nas caudas das distribuições. A remoção desses dados ruidosos não se mostra uma boa solução pois existem distribuições que seguem a lei da potência apenas na cauda.

Uma alternativa para tratar esses ruídos seria mexer no tamanho dos bins dos histogramas e normalizar a quantidade das amostras pelo tamanho dos bins. Isso pode auxiliar na suavização da curva pois aumentar o tamanho dos bins para as amostras na cauda acaba reduzindo os erros estatísticos, uma vez que eles acabam recebendo mais amostras. A maneira mais comum de ajustar os tamanhos dos bins é seguindo uma escala logarítmica (logarithmic binning).

Outra maneira de tratar esses ruídos, e que o autor afirma ser melhor, é calculando a função de distribuição cumulativa (cumulative distribution function). Ao invés de gerar o histograma dos dados, gera-se o gráfico da probabilidade de x ser maior ou igual à x. Caso a distribuição dos dados seja um distribuição de lei de potência, a função de distribuição cumulativa também terá uma distribuição de lei de potência, porém com um expoente de potência subtraído de uma unidade. Ao gerar o gráfico, também será obtida uma linha reta, porém com uma declividade reduzida.Algumas vantagens dessa abordagem é que elimina a tarefa de ter que decidir quais os tamanhos dos bins e não há perda de informação. Segundo estudos recentes, uma distribuição de lei de potência é sinônimo de distribuição de Pareto e de lei de Zipf.

Em situações em que os dados não foram gerados artificialmente, é importante saber o valor do coeficiente de potência (alfa). O método mais utilizado para calcular este coeficiente é através do declive da reta, porém ele pode introduzir vieses ao valor de alfa. Uma maneira alternativa de calcular o coeficiente de potência é através da equação abaixo.

$$\alpha = 1 + n[\sum_{i=1}^{n} ln\frac{x_i}{x_{min}}]^{-1}$$

É importante ressaltar que nem toda distribuição que apresenta uma característica enviesada necessariamente obedece a uma distribuição de lei de potência.

0s conclusão: 20:39