Prova

July 23, 2020

Atividade Avaliativa 01

Aluno: Kristtopher Kayo Coelho

Matrícula: ES95031

Bibliotecas

[2]:
```
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
import numpy as np
import pandas as pd
import statsmodels.api as sm
import math
from sklearn import linear_model
```

1 (10 pontos) Para uma v.a. que assume os valores

[3]:
```
X = [2, 10, 1, 2, 7, 6, 7, 8, 9, 1, 10, 5, 7, 9, 1, 3, 1]
```

a) (4 pontos) Caracterize esses dados, da melhor forma possível. Utilize métricas e distribuições que aprendeu durante a disciplina. Discuta a presença de outliers.

[4]:
```
print('-------Amostra X-------')
print('A amostra é composta por '+ str(len(X))+' valores.')
print('Média = {:0.2f}'.format(np.mean(X)))
print('Mediana = ',np.median(X))
print('Moda = ',np.argmax(np.bincount(X)))
print('Variâcia = {:0.2f}'.format(np.var(X)))
print('Desvio padrão = {:0.2f}'.format(np.std(X)))
print('Coeficiente de variação = {:0.2f}'.format(np.std(X)/np.mean(X)))
print('Valor máximo',max(X))
print('Valor mínimo',min(X))
```

```
-------Amostra X-------
A amostra é composta por 17 valores.
Média = 5.24
```

```
Mediana =  6.0
Moda =  1
Variâcia = 11.12
Desvio padrão = 3.33
Coeficiente de variação = 0.64
Valor máximo 10
Valor mínimo 1
```

A amostra é composta por uma variavel aletória discreta contendo 17 valores. Estes valores pertencentem ao conjunto dos números inteiros positivos compreendidos entre 1 e 10.

O dado que mais se repete na amostra é o 1.

O valor médio obtido entre os dados é de 5.24, próximo ao valor da mediana que é 6, o que sugere que os dados comparados possivelmente seguem uma distribuição uniforme.

O desvio padrão oponta para uma dispersão dos dados da amostra em 3.33.

A dispersão estatística apontada pela variância é de 11.12.

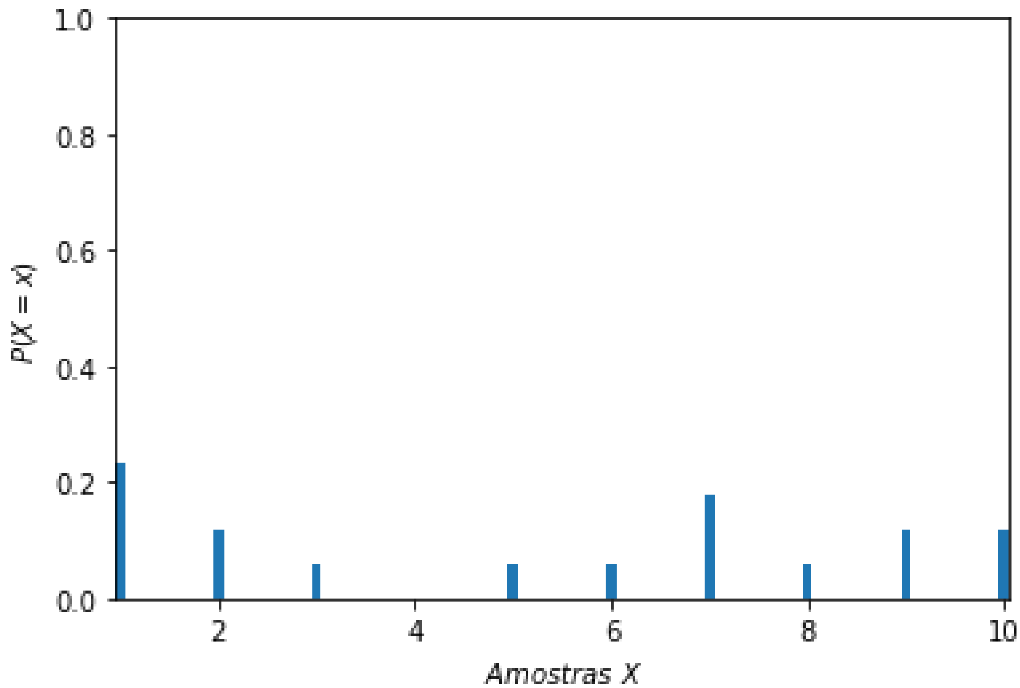
Além disso a amostra possui um coeficiente de variação elevado, 0.64 indicando a heterogeneidade da amostra.

A amostra não contém a presença de *outliers*.

b) (3 pontos) Faça a PMF.

[5]:
```python
df_amostra = pd.DataFrame({'':X})
ax = df_amostra.plot(kind='hist', density=True, histtype='bar', rwidth=0.1,
                     xlim=(min(X)-0.05, max(X)+0.05), ylim=(0,1),
                     legend=False, bins=np.arange(len(X))-0.5)
ax.set_xlabel(r'$Amostras~X$')
ax.set_ylabel(r'$P(X = x)$')
```

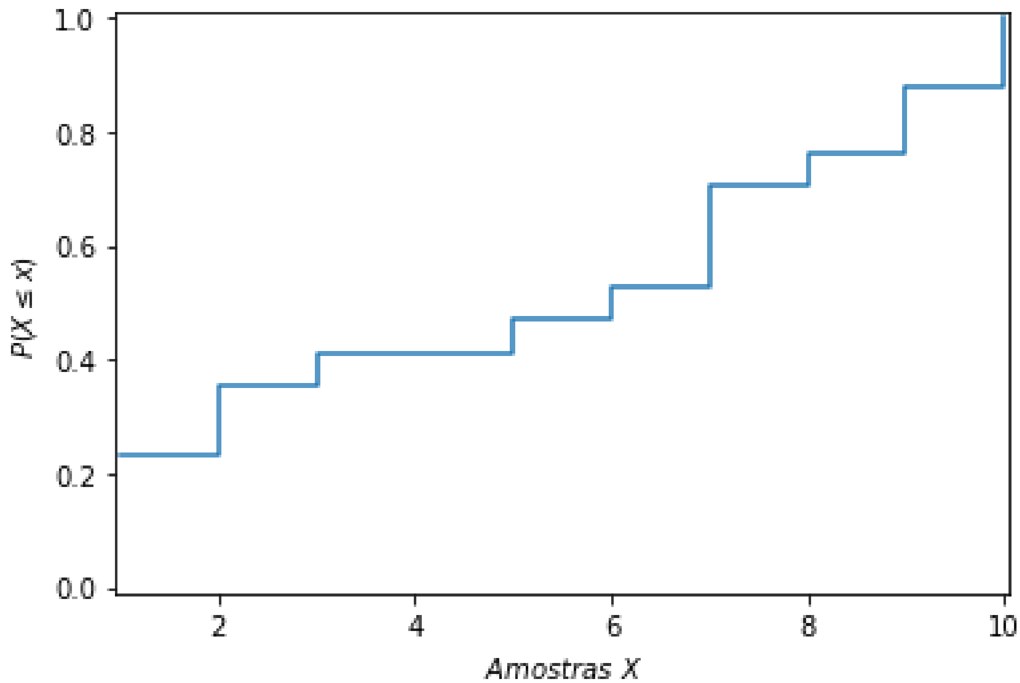
[5]:
```
Text(0, 0.5, '$P(X = x)$')
```



A função massa de probabilidade (PMF) associa um valor de probabilidade à cada possível ocorrência de uma variável aleatória discreta. Deste modo é possível destacar que a maior probabilidade de ocorrer o valor 1 com 0,23%, em seguida o vaor 7 com 0,17%.

c) (3 pontos) Faça a CDF.

[6]:
```
x = np.sort(X)
ecdf = sm.distributions.ECDF(X)
y = ecdf(x)

plt.step(x, y, where='post')
plt.ylabel(r'$P(X \leq x)$')
plt.xlabel(r'$Amostras~X$')
plt.xlim(min(x)-0.05, max(x)+0.05)
plt.ylim(0-0.01, 1+0.01)
plt.show()
```



De modo a análogo a PMF, a Função de Distribuição Cumulativa (CDF), também representa a probabilidade de ocorrencia de cada dado na amostra, portanto é observável uma maior porcentagem para os valores 1 e 7. Além disso o platô para o valor 4 no eixo x representa sua ausência na amostra.

2 (10 pontos) Considerando a seguinte amostra gerada aleatoriamente

X = np.random.normal(media, dp, 36)

X = np.around(X,2)

print(X)

[7]:
```
X = [2.57, 3.36, 5.28, 3.5, 2.93, 2.89, 0.9, 2.48, 3.37, 2.05, 2.77, 3.48, 2.4,␣
↪2.37,
    2.5, 1.36, 3.77, 1.74, 3.38, 1.94, 4.64, 4.08, 3.56, 3.28, 3.35, 4.46, 2.
↪06, 3.,
    3.91, 4.11, 1.53, 2.06, 2.78, 2.75, 1.14, 4.09]
```

a) (1 pontos) Calcule a média e o desvio padrão.

[8]:
```
#media = np.mean(X) com o passo a passo
x = 0
```

```python
for i in X:
    x += i
media = x/len(X)
print('Média = {:0.2f}'.format(media))
```

```
Média = 2.94
```

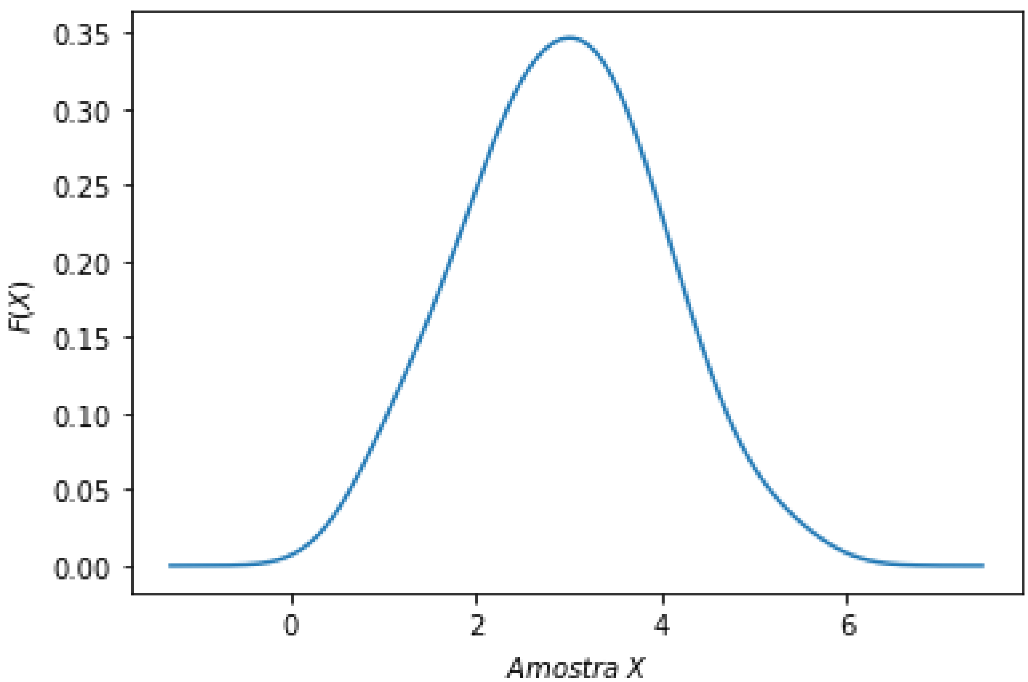
[9]:
```python
#desvio padrão  S = np.std(X) com o passo a passo
S = math.sqrt(np.mean(abs(X - np.mean(X))**2))
print('Desvio padrão = {:0.2f}'.format(S))
```

```
Desvio padrão = 1.00
```

b) (3 pontos) Plote a PDF

[10]:
```python
ser = pd.Series(X)
ser.plot.kde()
plt.ylabel(r'$F(X)$')
plt.xlabel(r'$Amostra~X$')
```

[10]: `Text(0.5, 0, '$Amostra~X$')`



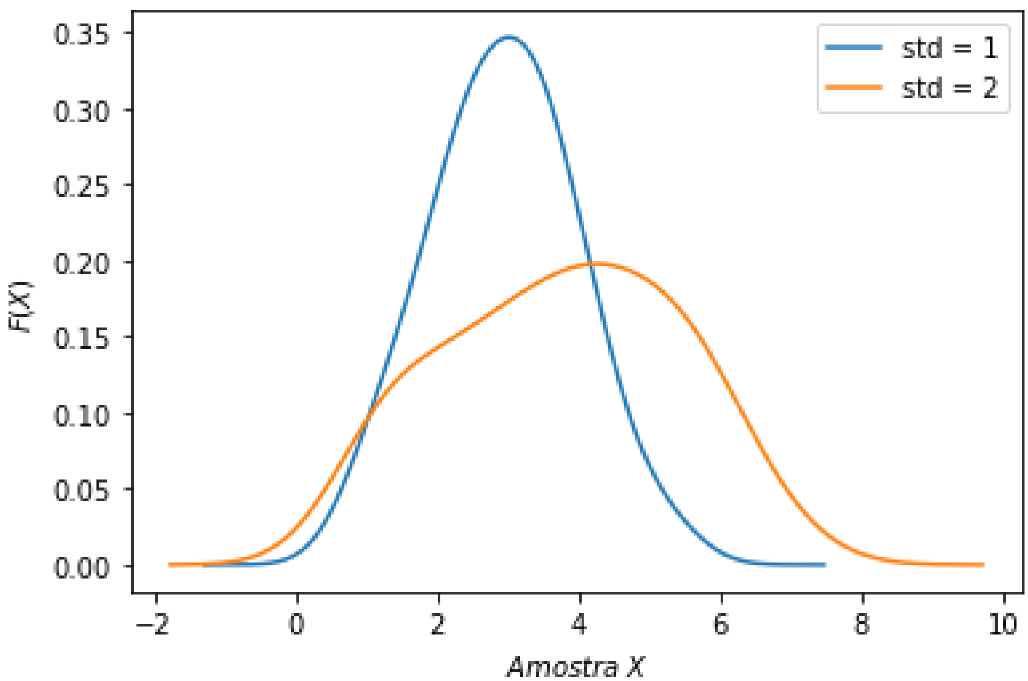
A função de densidade de probabilidade aponta para maior probabilidade de os valores da variável aleatória serem próximos à média, afastando-se dela em 1 unidade de medida, para mais ou para menos.

c) (2 pontos) Plote a PDF, no mesmo gráfico, considerando a mesma média e um desvio padrão igual a 2 (dica: não se preocupe com o eixo Y).

Para esta questão optou-se por gerar uma nova amostra com os mesmos padrões, media = 3 e n = 36, variando apenas o desvio padrão de 1 para 2.

[11]:
```
X_new = np.random.normal(round(media), 2, 36)
X_new = np.around(X_new,2)

df = pd.DataFrame({'std = 1': X,'std = 2': X_new,})
df.plot.kde()
plt.ylabel(r'$F(X)$')
plt.xlabel(r'$Amostra~X$')
```

[11]:
```
Text(0.5, 0, '$Amostra~X$')
```



Para uma amostra do mesmo tamanho e mantendo a mesma média e distribuição, pode-se observar que ao aumentar o desvio padrão ocorre um achatamento da curva. Portanto existe uma maior probabilidade dos dados se afastarem da média em um fator de 2 unidades para mais ou para menos.

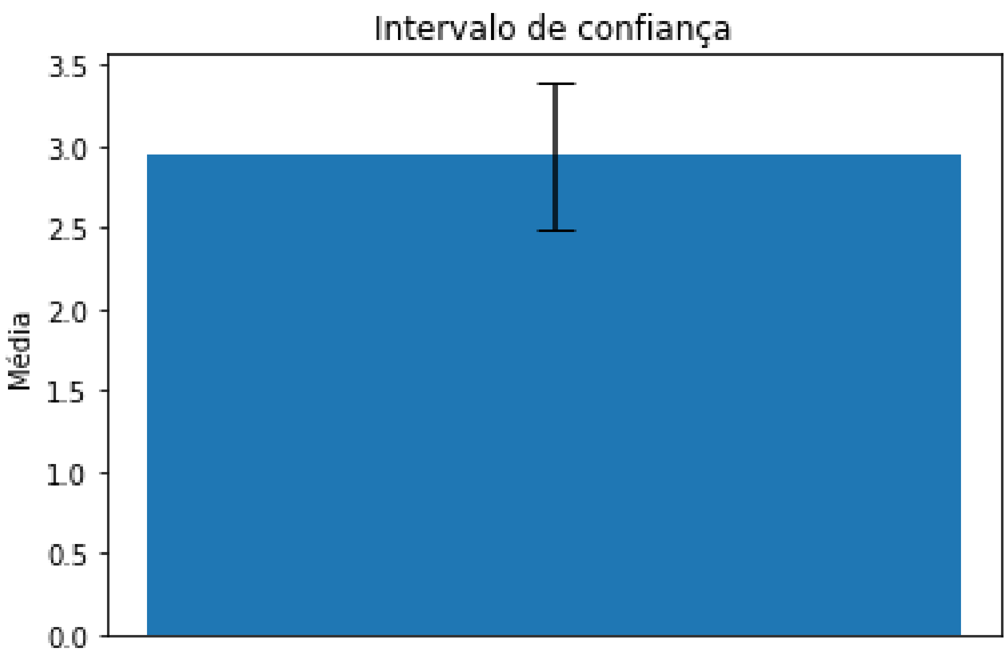
d) (4 pontos) Calcule o intervalo de confiança para a média com um nível de confiança de 99% (dica: considerar média e desvio padrão calculados em "a)").

[12]:
```
t = stats.t.ppf((0.99 + 1)/2, len(X))
A_min = media - t * (S/(math.sqrt(len(X))))
```

```
A_max = media + t * (S/(math.sqrt(len(X))))
print('O intervalo de confiança para a média com um nível de confiança de 99% é␣
 ↪entre {:0.4f} e {:0.4f}'.format(A_min, A_max))

plt.bar(0, media, yerr = (media - A_min), capsize=7)
plt.ylabel('Média')
plt.title('Intervalo de confiança')
plt.xticks([])
```

```
O intervalo de confiança para a média com um nível de confiança de 99% é entre
2.4865 e 3.3935
```

[12]:
```
([], <a list of 0 Text major ticklabel objects>)
```



3 (10 pontos) Refaça de (a) a (d) do exercício (2) com novos dados gerados aleatoriamente e discuta a semelhança com os dados que estão disponíveis nessa prova.

[13]:
```
Y = np.random.normal(round(media), S, 36)
Y = np.around(Y,2)
print(Y)
```

```
[4.76 3.09 2.45 4.34 2.36 2.58 3.5  2.01 4.34 2.12 2.09 3.17 2.45 2.97
```

```
 3.8  0.99 3.79 2.11 2.99 1.9  3.89 2.61 4.18 3.36 2.21 2.24 2.4  2.27
 1.69 2.48 3.1  2.68 2.11 1.38 2.98 3.32]
```

a) (1 pontos) Calcule a média e o desvio padrão.

[14]:
```
y_media = np.mean(Y)
print('Média = {:0.2f}'.format(y_media))
```

```
Média = 2.80
```

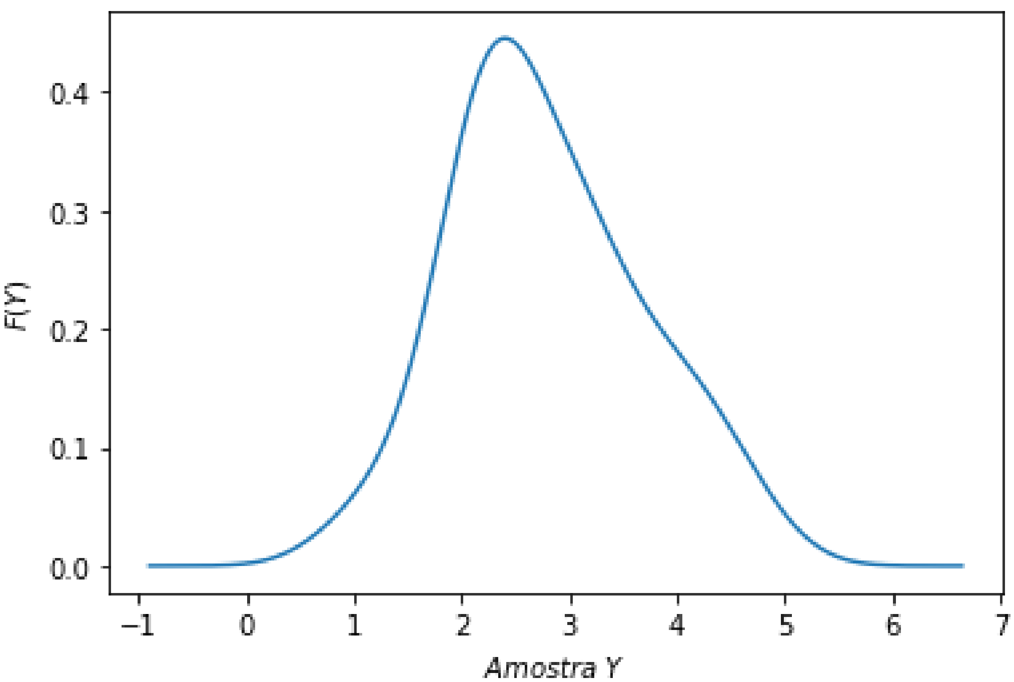
[15]:
```
y_S = np.std(Y)
print('Desvio padrão = {:0.2f}'.format(y_S))
```

```
Desvio padrão = 0.86
```

b) (3 pontos) Plote a PDF

[16]:
```
ser = pd.Series(Y)
ser.plot.kde()
plt.ylabel(r'$F(Y)$')
plt.xlabel(r'$Amostra~Y$')
```

[16]:
```
Text(0.5, 0, '$Amostra~Y$')
```

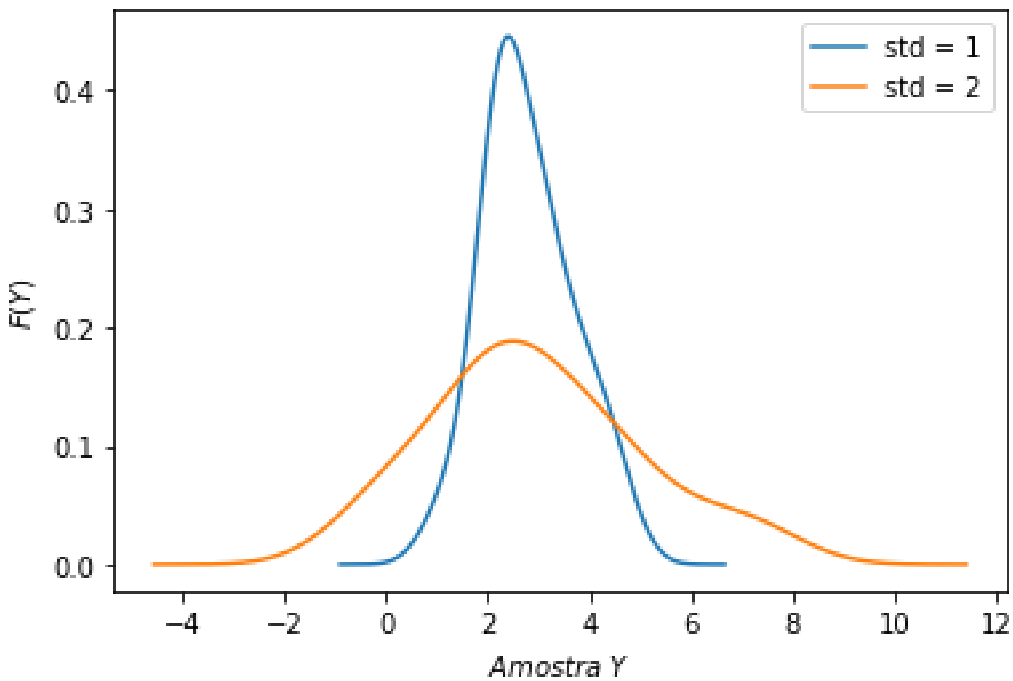


A função de densidade de probabilidade aponta para maior probabilidade de os valores da variável aleatória serem próximos à média, afastando-se dela em 1 unidade de medida, para mais ou para menos.

c) (2 pontos) Plote a PDF, no mesmo gráfico, considerando a mesma média e um desvio padrão igual a 2 (dica: não se preocupe com o eixo Y).

[17]:
```
Y_new = np.random.normal(round(y_media), 2, 36)
Y_new = np.around(Y_new,2)

df = pd.DataFrame({'std = 1': Y,'std = 2': Y_new,})
df.plot.kde()
plt.ylabel(r'$F(Y)$')
plt.xlabel(r'$Amostra~Y$')
```

[17]: Text(0.5, 0, '$Amostra~Y$')



Para uma amostra do mesmo tamanho e mantendo a mesma média e distribuição, pode-se observar que ao aumentar o desvio padrão ocorre um achatamento da curva. Portanto existe uma maior probabilidade dos dados se afastarem da média em um fator de 2 unidades para mais ou para menos.

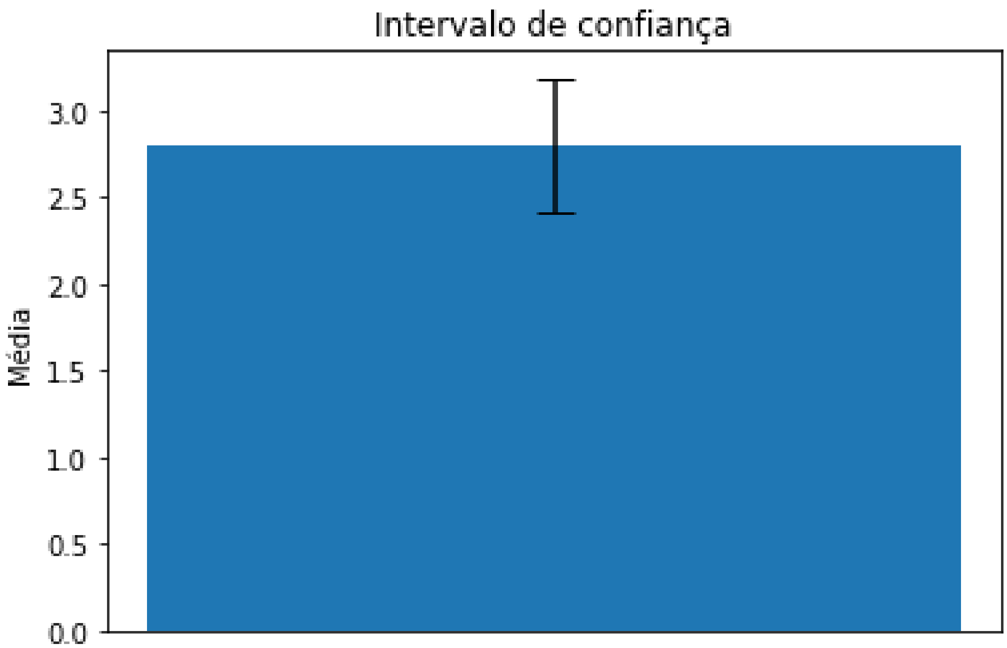
d) (4 pontos) Calcule o intervalo de confiança para a média com um nível de confiança de 99% (dica: considerar média e desvio padrão calculados em "a)").

[18]:
```
t = stats.t.ppf((0.99 + 1)/2, len(Y))
Ay_min = y_media - t * (y_S/(math.sqrt(len(Y))))
Ay_max = y_media + t * (y_S/(math.sqrt(len(Y))))
print('O intervalo de confiança para a média com um nível de confiança de 99% é␣
↪entre {:0.4f} e {:0.4f}'.format(A_min, A_max))
```

```
plt.bar(0, y_media, yerr = (y_media - Ay_min), capsize=7)
plt.title('Intervalo de confiança')
plt.ylabel('Média')
plt.xticks([])
```

```
O intervalo de confiança para a média com um nível de confiança de 99% é entre
2.4865 e 3.3935
```

```
[18]: ([], <a list of 0 Text major ticklabel objects>)
```



Ao realizar o exercício 3, repetindo a dinâmica do exercício 2 para novos dados gerados aleatoriamente, observa-se que, ao manter os mesmos parametros para a geração dos dados, ou seja, utilizando a distribuição normal, média 3 e desvio padrão 1, os resultados foram extremamente semelhantes. Tais como para a variação do desvio padrão para valor 2.

Isto indica que, para todas as amostras que forem geradas com estas características, teremos a mairoia dos dados próximos da média 3 e desvio padrão semelhantes, bem como as demais métricas estatísticas.

Nota-se ainda que a variação de quaisquer parametro tem grande influência nos resultados das amostras, haja visto os gráficos apresentados com desvio padrão (std) igual a 2.

4 (10 Pontos) O tempo de processamento necessário para executar uma tarefa foi medido em dois sistemas. Esses sistemas são significativamente diferentes num nível de confiança (NC) de 95 porcento?

Sistema A: { 3.38, 2.50, 3.52, 19.12, 3.60, 1.74}

Sistema B: {0.64, 0.62, 7.26, 5.36, 16.57, 1.41}.

Funções de cálculo e análise do intervalo e plotagem do gráfico para ambos os sistemas.

[19]:
```
def confidence_interval_analysis(A_min, A_max, n0_mean, B_min, B_max, n1_mean,␣
↪interval):
    print('Conclusão:')
    if((n1_mean >= A_min) and (n1_mean <= A_max)) and ((n0_mean >= B_min) and␣
↪(n0_mean <= B_max)):
        print('As Amostras não são diferentes neste nível de confiança de {:0.
↪1f}%.'.format(interval*100))
        return
    elif(((n0_mean > n1_mean) and (A_min > B_max)) or((n1_mean > n0_mean) and␣
↪(B_min > A_max))) :
        print('As Amostras são diferentes neste nível de confiança de {:0.1f}%.
↪'.format(interval*100))
    else:
        print ('Inconclusivo pata o T-student, o teste T é necessário nível de␣
↪confiança de {:0.1f}%.'.format(interval*100))
```

[20]:
```
def confidence_interval(interval, mean, std, n):
    t = stats.t.ppf((interval + 1)/2, n)
    return (mean - t*(std/(math.sqrt(n)))), (mean + t*(std/(math.sqrt(n))))
```

[21]:
```
def graph(n0_mean, n1_mean, n0_interval, n1_interval):
    x = [0, 1]
    y =  [n0_mean, n1_mean]
    error = [n0_interval, n1_interval]
    labels = ['Sistema A', 'Sistema B']

    fig, ax = plt.subplots()
    ax.bar(x, y,
           yerr=error,
           align='center',
           alpha=0.5,
           ecolor='black',
           capsize=10)
    ax.set_ylabel('Média ($t$)')
    ax.set_xticks(x)
    ax.set_xticklabels(labels)
```

[22]:
```
Sistema_A = [3.38, 2.50, 3.52, 19.12, 3.60, 1.74]
Sistema_B = [0.64, 0.62, 7.26, 5.36, 16.57, 1.41]
# Sistema_A = [3.38, 2.50, 3.52, 3.60, 1.74]
# Sistema_B = [0.64, 0.62, 7.26, 5.36, 1.41]
```

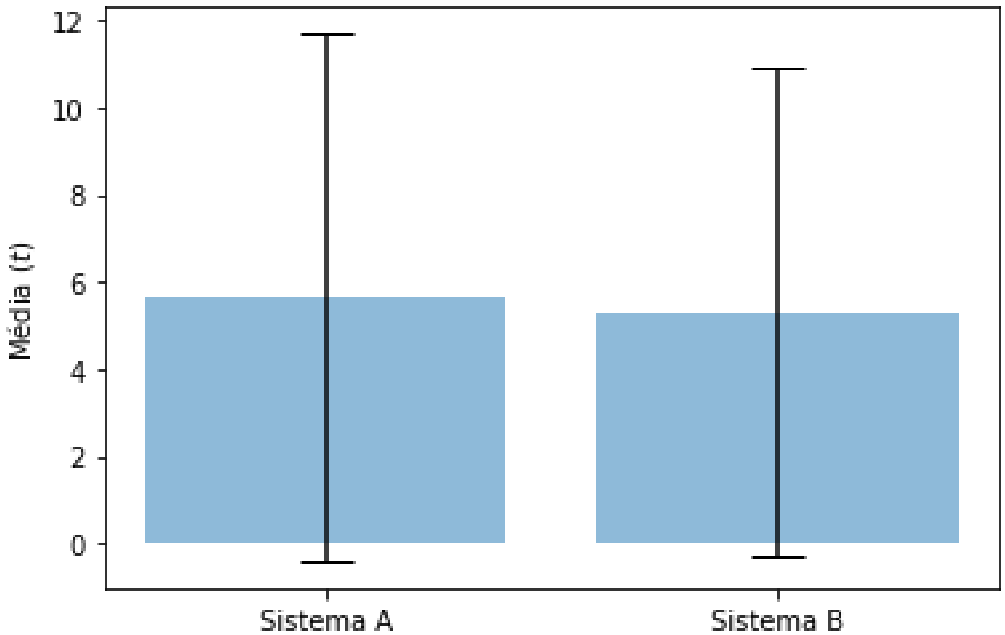
Teste T-student

[23]:
```
t, p = stats.ttest_ind(Sistema_A, Sistema_B)
print('Existem {:0.2f}% de chances da diferença ser ao acaso H0.'.format(p*100))
```

```
Existem 92.99% de chances da diferença ser ao acaso H0.
```

[24]:
```
A_min, A_max = confidence_interval(0.95, np.mean(Sistema_A), np.std(Sistema_A),␣
↪len(Sistema_A))
B_min, B_max = confidence_interval(0.95, np.mean(Sistema_B), np.std(Sistema_B),␣
↪len(Sistema_B))

print('Intervalo de confiança da amostra A: {:0.2f}, {:0.2f} e média {:0.2f}.'.
↪format(A_min, A_max, np.mean(Sistema_A)))
print('Intervalo de confiança da amostra B: {:0.2f}, {:0.2f} e média {:0.2f}.
↪\n'.format(B_min, B_max, np.mean(Sistema_B)))

confidence_interval_analysis(A_min, A_max, np.mean(Sistema_A), B_min, B_max, np.
↪mean(Sistema_B), 0.95)

graph(np.mean(Sistema_A), np.mean(Sistema_B), (A_max - np.mean(Sistema_A)),␣
↪(B_max - np.mean(Sistema_B)))
```

```
Intervalo de confiança da amostra A: -0.41, 11.70 e média 5.64.
Intervalo de confiança da amostra B: -0.31, 10.93 e média 5.31.

Conclusão:
As Amostras não são diferentes neste nível de confiança de 95.0%.
```



Existe a presença de *outliers* em ambas as amostras os que faz com que o desvio padrão seja alto, cerca de 6 (unidades de tempo), porém mesmo com a remoção dos destes valores destoantes as amostras não são diferentes entre si para o nível de confiança de 95.0%.

5 (5 pontos) A partir dos dados a seguir:

xi = [1, 2, 3, 4, 5, 6, 7, 8]

yi = [0.3, 0.5, 1.2, 1.4, 1.8, 3.2, 4, 2.6]

yi = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.2]

ei = [0.7, 1, 0.8, 1.1, 1.2, 0.3, 0, 1.6]

a) (2 pontos) Faça o teste visual de linearidade e analise-o.

Como a interpretação da questão faz parte da avaliação, e esta gerou redundancia de interpretações, optou-se por apresentar o teste visual de linearidade de duas maneiras. A primeira, mais tradicional, para regressão linear simples e em seguida a regressão multipla.

Para a primeira interpretação, mais tradicional, é considerado os dados da seguinte maneira:

xi = [1, 2, 3, 4, 5, 6, 7, 8]

yi = [0.3, 0.5, 1.2, 1.4, 1.8, 3.2, 4, 2.6]

ŷi = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.2]

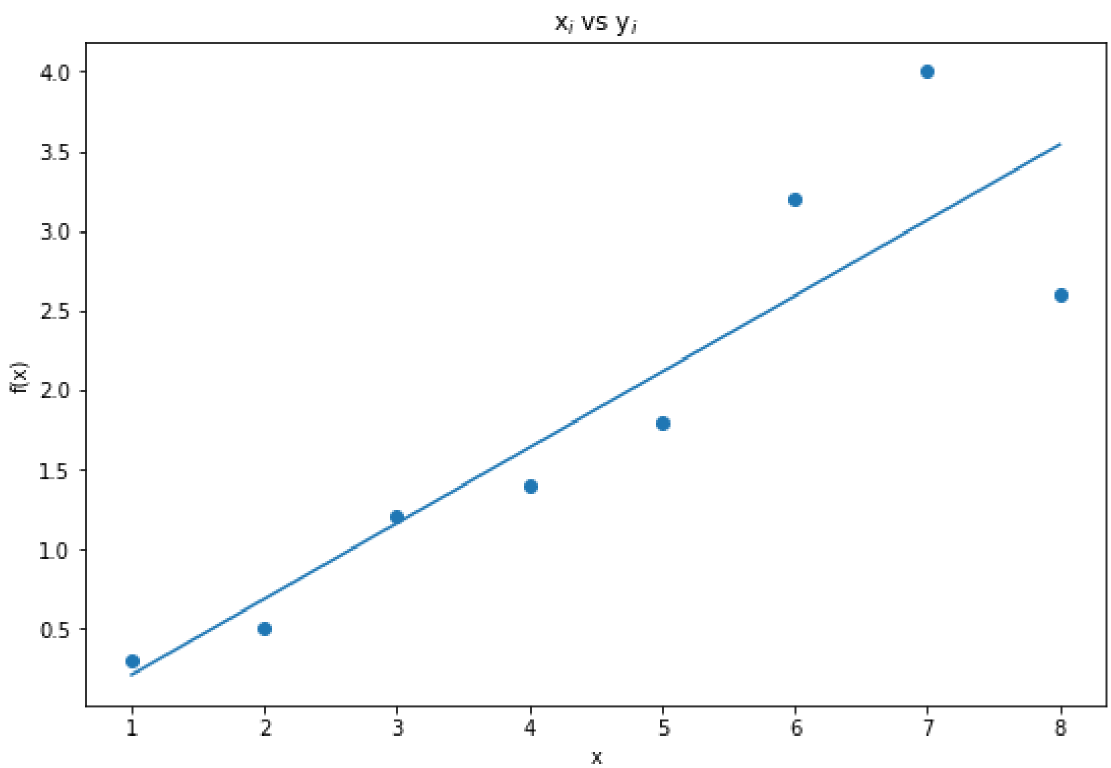
ei = [0.7, 1, 0.8, 1.1, 1.2, 0.3, 0, 1.6]

Deste modo é realizada a regração linear simples.

O gráfico abaixo representa o teste visual de linearidade para os dados de x_i e y medido (y_i)

[25]:
```
figure = plt.figure(figsize=(9,6))
plt.scatter(df['xi'], df['y0'])
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('x$_i$ vs y$_i$')
m, b = np.polyfit(df['xi'], df['y0'], 1)
plt.plot(df['xi'], m*df['xi'] + b)
```

[25]:
```
[<matplotlib.lines.Line2D at 0x7f004e75dad0>]
```



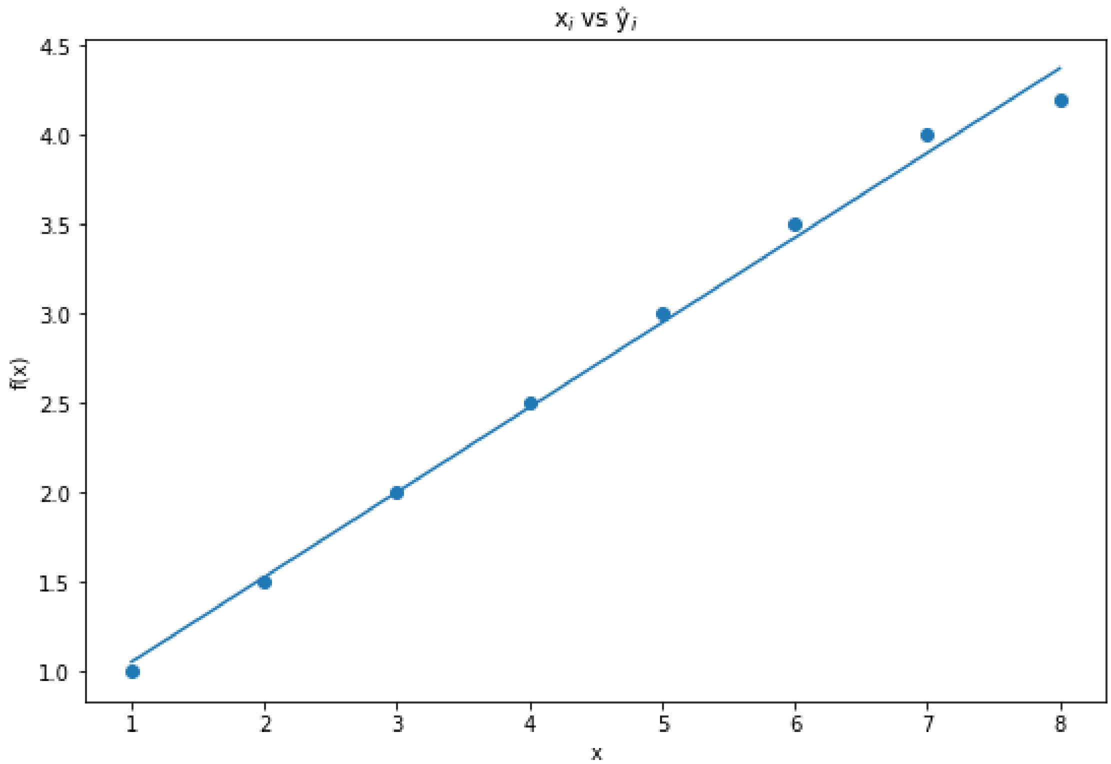
Conclusão: De acordo com o Teste Visual de Pressupostos para linearidade, a curva (x_i,y_i) tende a ser linear. Os resultados ajustados se aproximam de uma reta normal onde a curva pode ser descrita pela equação $y = ax + b + e$.

O próxmo gráfico representa o teste visual de linearidade para os dados de x_i e y calculado ($\hat{y}_i$)

[26]:
```
figure = plt.figure(figsize=(9,6))
plt.scatter(df['xi'], df['y1'])
plt.xlabel('x')
plt.ylabel('f(x)')
```

```
plt.title('x$_i$ vs ŷ$_i$')
m, b = np.polyfit(df['xi'], df['y1'], 1)
plt.plot(df['xi'], m*df['xi'] + b)
```

[26]: [<matplotlib.lines.Line2D at 0x7f004e6d28d0>]



Conclusão: De modo similar, também de acordo com o Teste Visual de Pressupostos para linearidade, a curva dos erros (x_i,$\hat{y}_i$) tende a ser linear. Os resultados ajustados se aproximam de uma reta normal onde a curva pode ser descrita pela equação $y = ax + b + e$.

Deste modo é possível concluir que estes dados foram calculados a partir da regressão linear obtida para os dados $[x_i, y_i]$.

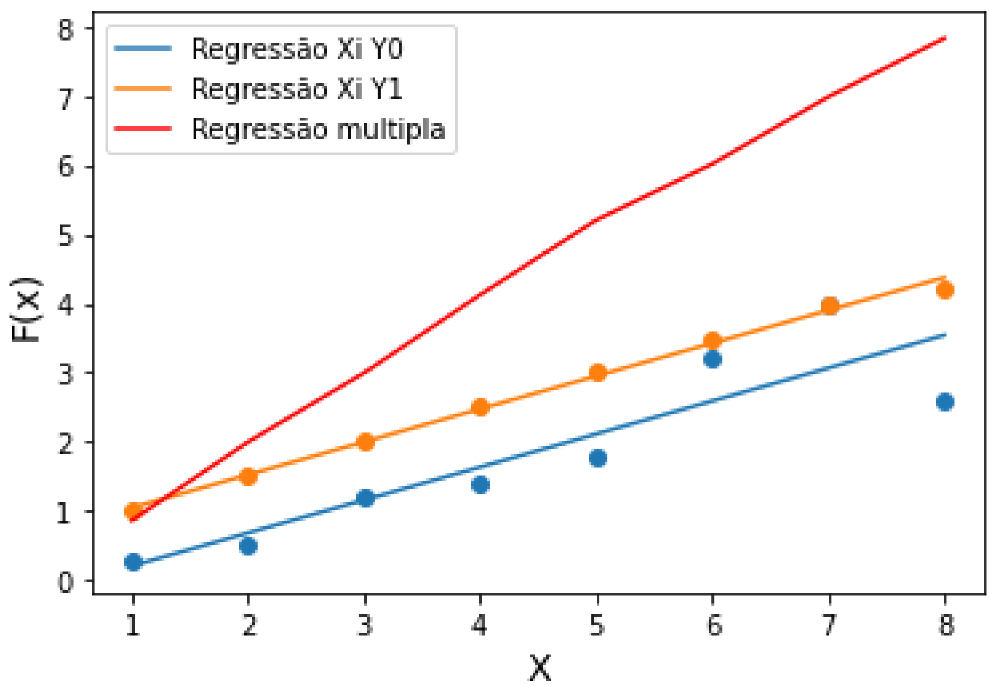
À seguir é apresentado a regrssão linear multipla, considerando os dados (x_i,[y_0,y_1]).

[26]:
```
data = {'xi': [1, 2, 3, 4, 5, 6, 7, 8],
                'y0': [0.3, 0.5, 1.2, 1.4, 1.8, 3.2, 4, 2.6],
                'y1': [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.2],
                'ei': [0.7, 1, 0.8, 1.1, 1.2, 0.3, 0, 1.6]}

df = pd.DataFrame(data,columns=['xi','y0','y1','ei'])

X = df['xi']
```

```python
Y = df[['y0','y1']]

#regressão linear multipla
regr = linear_model.LinearRegression()
regr.fit(Y, X)
a = regr.intercept_
c = regr.coef_
print('Onde: b0 =',a,' b1 =',c[0], ' b2 =',c[1])
#Equação da regressão linear multipla
eq = a + c[0]*df['y0'] + c[1]*df['y1'] #+ df['ei']

#Apresentação dos dados em forma gráfica
plt.scatter(df['xi'], df['y0'])
m, b = np.polyfit(df['xi'], df['y0'], 1)
plt.plot(df['xi'], (m*np.array(df['xi']) + b),label = 'Regressão Xi Y0')

plt.scatter(df['xi'], df['y1'])
m, b = np.polyfit(df['xi'], df['y1'], 1)
plt.plot(df['xi'], (m*np.array(df['xi']) + b),label = 'Regressão Xi Y1')

plt.plot(df['xi'], eq, color = 'red', label = 'Regressão multipla')

plt.xlabel('X', fontsize=14)
plt.ylabel('F(x)', fontsize=14)
plt.grid(False)
plt.legend(loc = 'best')
```

```
Onde: b0 = -1.4206896202223716  b1 = -0.265601086526864  b2 = 2.3663379382341905
```

```
[26]: <matplotlib.legend.Legend at 0x7ffb99c2cb90>
```



Para esta segunda interpretação, as linhas de cores azul e laranja apresentam regressões lineares individuais para cada par (x_i,y_i). Onde laranja representa (x_i,y_0) e azul representa representa (x_i,y_1).

Em vermelho é apresentado a regressão linear multipla para (x_i,[y_0,y_1])

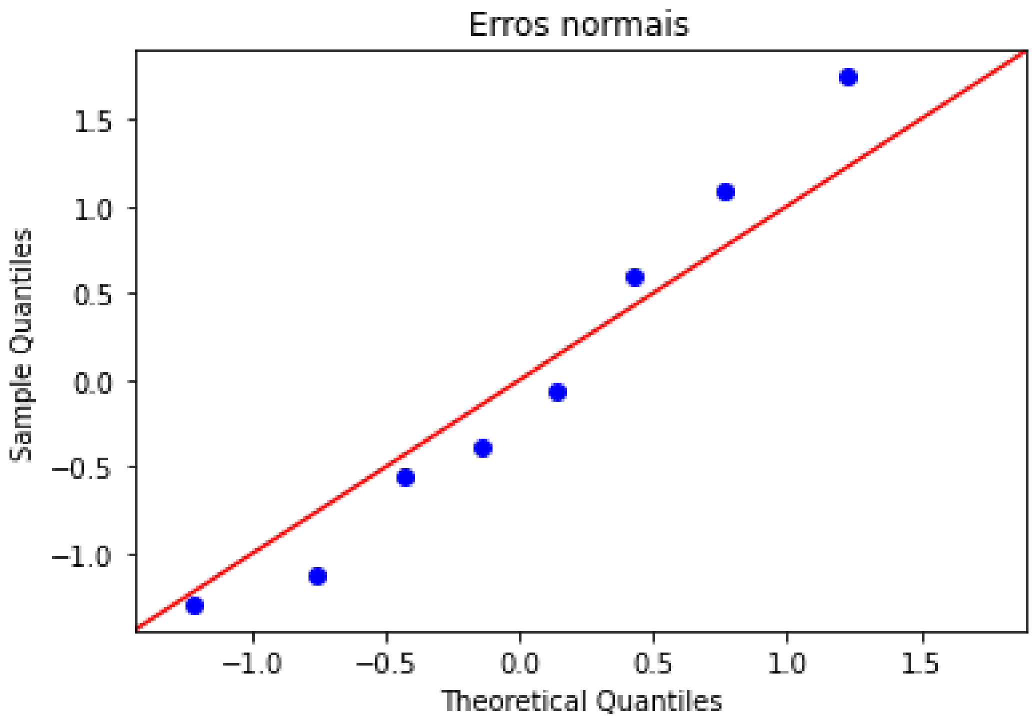
Conclusão: De acordo com o Teste Visual de Pressupostos para linearidade, ambas curvas tendem a serem lineares. Os resultados ajustados se aproximam de uma reta normal para cada par (x_i,y_i). Portanto diz-se que os erros são normais e curva pode ser descrita pela equação $y = ax + b + e$. Para a regressão multipla (x_i,[y_0,y_1]) temos a seguinte equação: $y = b_0 + b_1x_1 + b_2x_2 + e$

b) (3 pontos) Faça os testes visuais de erros independentes e homocedasticidade e analise-os.

Antes, foi realizado o teste dos erros normais, os demais aparecem a seguir.

[27]:
```
sm.qqplot(df['y0'],line = '45', fit = True)
plt.title('Erros normais')
```

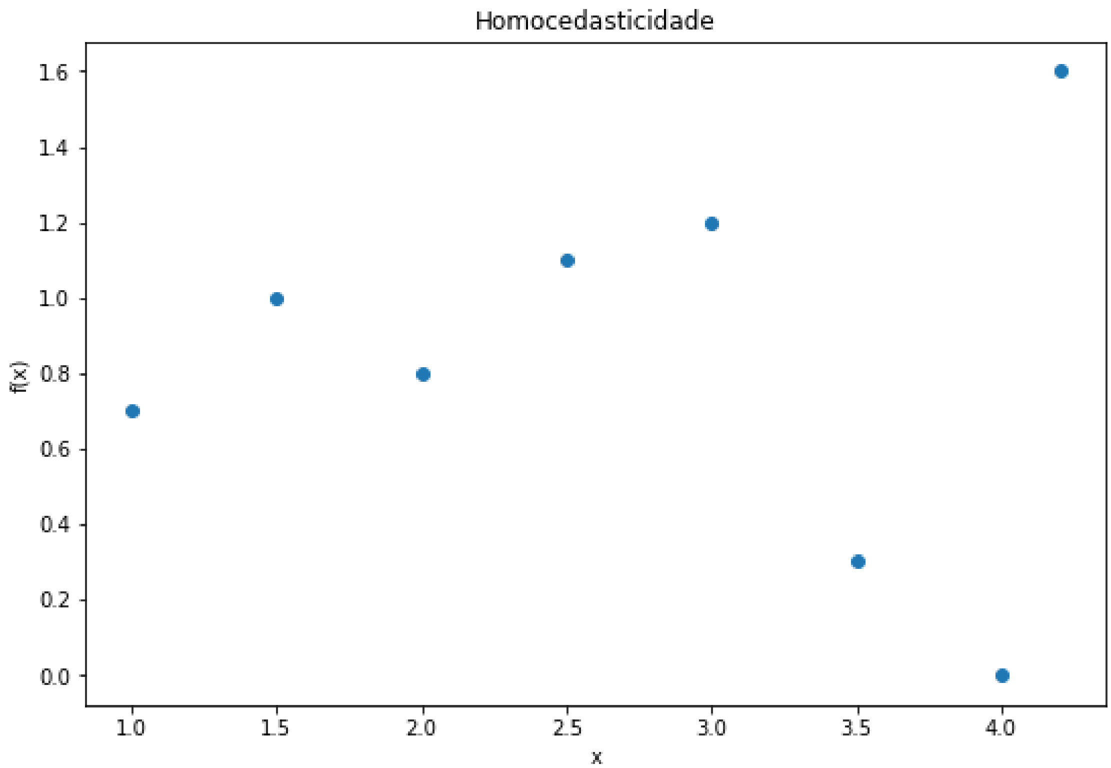
[27]: `Text(0.5, 1.0, 'Erros normais')`



Os pontos se aproximam da reta da normal, portanto diz-se que os erros são normais.

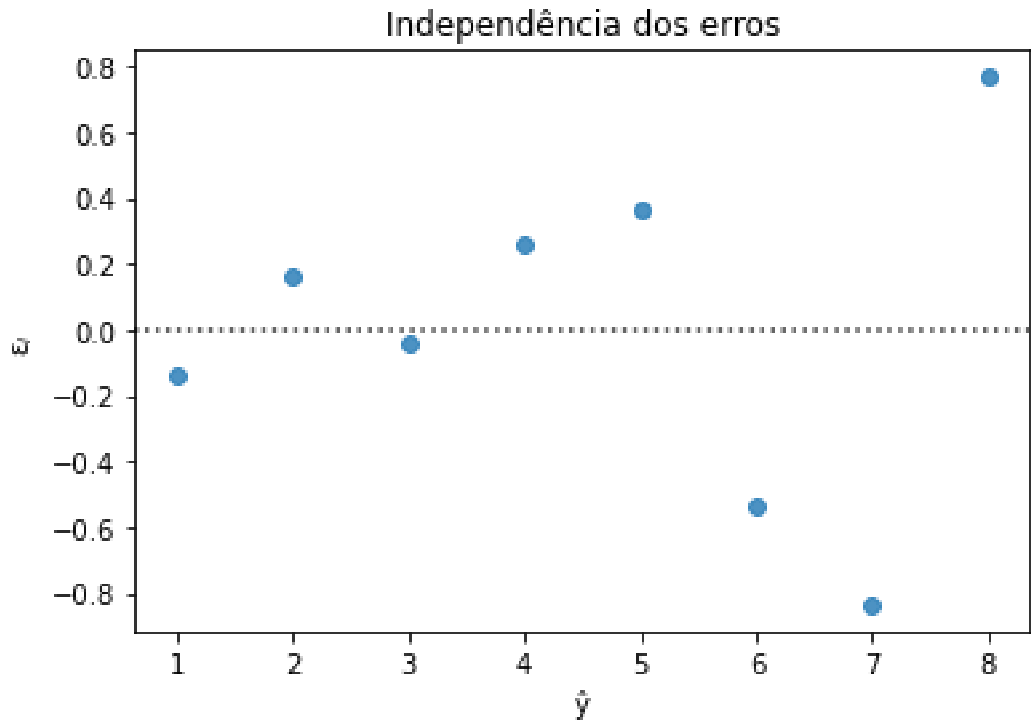
[28]:
```
figure = plt.figure(figsize=(9,6))
plt.scatter(df['y1'], df['ei'])
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Homocedasticidade')
# m, b = np.polyfit(df['y1'], df['ei'], 1)
# plt.plot(df['y1'], m*df['y1'] + b)
```

[28]: Text(0.5, 1.0, 'Homocedasticidade')



[29]:
```
sns.residplot(df['xi'],df['ei'])
plt.xlabel('ŷ')
plt.ylabel(' $_i$')
plt.title('Independência dos erros')
```

[29]: Text(0.5, 1.0, 'Independência dos erros')



Conclusão: Para os testes de independência dos erros e homocedasticidade, percebe-se que exite uma tendência ao espalhamento enquanto o valor de $\hat{y}$ cresce. Portanto sugere-se usar regressão não-linear ou linearização.