

Medidas de Posição e Dispersão

Disciplina: Estatística Aplicada a Engenharia de Software

Prof. Me. Max Gabriel Steiner

Introdução – Medidas de Posição e Dispersão

- Nessa etapa vamos trabalhar com as medidas de posição e com as medidas de dispersão.
- Medidas de posição: vão indicar aonde estão os números dentro de uma distribuição.
- Medidas de Dispersão: vão indicar como os dados estão dispersos na base de dados.
- Este assunto faz parte da estatística descritiva – que visa descrever e sumarizar um conjunto de dados. Portanto:
- Vamos trabalhar com vários conceitos como por exemplo:
- Média, mediana, moda, média aritmética, geométrica, harmônica e quadrática.
- Fechamos as medidas de posição com o estudo dos quartis e percentis.

Introdução – Medidas de Posição e Dispersão

- Em seguida, com as medidas de dispersão vamos trabalhar com:
- Variância, desvio padrão e coeficiente de variação.
- Por fim, vamos reunir estes conceitos e aplicar na linguagem de máquina através da: avaliação de algoritmos de machine learning e também seleção de atributos com variância (visando selecionar os melhores atributos em uma base de dados).

MÉDIA ARITMÉTICA, MODA E MEDIANA – DADOS NÃO AGRUPADOS

150	151	152	152	153	154	155	155	155	155
156	156	156	157	158	158	160	160	160	160
160	161	161	161	161	162	163	163	164	164
164	165	166	167	168	168	169	170	172	173

Média

$$\bar{x} = \frac{\sum x_i}{n}$$

160.375

Moda

160

Mediana (ímpar)

$$\text{Mediana} = \frac{n}{2}$$

$$\text{Mediana} = \frac{9}{2}$$

$$\text{Mediana} = 4,5$$

$$\text{Mediana} = 5 \text{ (arredondado)}$$

Mediana (par)

$$m = \frac{n}{2}$$

$$m = 20$$

$$m = \frac{160 + 160}{2}$$

$$m = 160$$

150	151	152	152	153	154	155	155	155
-----	-----	-----	-----	-----	-----	-----	-----	-----



MÉDIA ARITMÉTICA, MODA E MEDIANA – DADOS NÃO AGRUPADOS

150	151	152	152	153	154	155	155	155	155
156	156	156	157	158	158	160	160	160	160
160	161	161	161	161	162	163	163	164	164
164	165	166	999	168	168	900	170	172	173

Média

$$\bar{x} = \frac{\sum x_i}{n}$$

199.225

Moda

160

Mediana (par)

$$m = \frac{n}{2}$$

$$m = 20$$

$$m = \frac{160 + 160}{2}$$

$$m = 160$$



Fonte: https://www.corrediodiferença.com.br/app/uploads/2022/02/20/revista_ciencia_saude_754708/estudo-aporte-quem-pesou-afetar-seu-risco-de-desenvolver-cancer-28108

➤ Hora de irmos ao Python aplicar os conceitos!

▼ Medidas de posição e dispersão

▼ Base de dados

```
[210] import numpy as np
import statistics
from scipy import stats
import math
```

```
[106] dados = np.array([150, 151, 152, 152, 153, 154, 155, 155, 155, 155, 156, 156, 156,
                        157, 158, 158, 160, 160, 160, 160, 160, 161, 161, 161, 161, 162,
                        163, 163, 164, 164, 164, 165, 166, 167, 168, 168, 169, 170, 172,
                        173])
```

✓ Média aritmética simples

```
[231] dados.sum() / len(dados)
```

```
➡ 160.375
```

```
[232] dados.mean()
```

```
➡ 160.375
```

```
▶ statistics.mean(dados)
```

```
➡ 160
```

✓ Moda

```
[234] statistics.mode(dados)
```

```
➡ 160
```

```
▶ stats.mode(dados)
```

```
➡ ModeResult(mode=160, count=5)
```

✓ Mediana

```
▶ dados_impar = [150, 151, 152, 152, 153, 154, 155, 155, 155]
```


▼ Cálculo manual (ímpar)

```
[237] posicao = len(dados_impar) / 2  
      posicao
```

↩ 4.5

```
[238] posicao = math.ceil(posicao)  
      posicao
```

↩ 5

▶ dados_impar[posicao - 1]

↩ 153

▼ Cálculo manual (par)

```
[254] posicao = len(dados) // 2  
      posicao
```

↩ 20

```
[255] dados[posicao - 1], dados[posicao]
```

↩ (160, 160)

▶ mediana = (dados[posicao - 1] + dados[posicao]) / 2
 mediana

↩ 160.0

▼ Bibliotecas

```
[257] np.median(dados_impar)
```

```
↩ 153.0
```

```
[258] np.median(dados)
```

```
↩ 160.0
```

```
[259] statistics.median(dados_impar)
```

```
↩ 153
```

```
▶ statistics.median(dados)
```

```
↩ 160.0
```



MÉDIA ARITMÉTICA PONDERADA

Bimestre	Nota	Peso
1º	9	1
2º	8	2
3º	7	3
4º	3	4

$$\bar{x} = \frac{\sum x_i}{n}$$

$$\frac{9 + 8 + 7 + 3}{4} = 6,75$$

$$M_p = \frac{p_1 \cdot x_1 + p_2 \cdot x_2 + \dots + p_n \cdot x_n}{p_1 + p_2 + \dots + p_n}$$

$$\frac{9 * 1 + 8 * 2 + 7 * 3 + 3 * 4}{1 + 2 + 3 + 4} = 5,80$$

$$\frac{9 + 8 + 8 + 7 + 7 + 7 + 3 + 3 + 3 + 3}{1 + 2 + 3 + 4}$$

✓ Média aritmética ponderada

```
[261] notas = np.array([9, 8, 7, 3])  
      pesos = np.array([1, 2, 3, 4])
```

```
[262] (9 * 1 + 8 * 2 + 7 * 3 + 3 * 4) / (1 + 2 + 3 + 4)
```

↔ 5.8

```
[263] media_ponderada = (notas * pesos).sum() / pesos.sum()  
      media_ponderada
```

↔ 5.8

```
▶ np.average(notas, weights=pesos)
```

↔ 5.8

MÉDIA ARITMÉTICA, MODA E MEDIANA – DADOS AGRUPADOS

Estatura (cm)	f_i	x_i	$f_i \cdot x_i$	F_i
150 -- 154	5	152	760	5
154 -- 158	9	156	1404	14
158 -- 162	11	160	1760	25
162 -- 166	7	164	1148	32
166 -- 170	5	168	840	37
170 -- 174	3	172	516	40
Total	40		6428	

Ponto médio de uma classe (x_i)

$$X_i = (L_i + l_i) / 2 = (158 + 154) / 2 = 156 \text{ cm}$$

$$\frac{\sum f_i}{2} = \frac{40}{2} = 20$$

Média

$$\bar{x} = \frac{\sum f_i \cdot x_i}{\sum f_i}$$

$$\bar{x} = \frac{6428}{40}$$

$$\bar{x} = 160,7$$

Moda

160

$$Md = l + \frac{(\frac{\sum f_i}{2} - F_{ant}) \cdot h}{f_i}$$

$$Md = 158 + \frac{(20 - 14) \cdot 4}{11}$$

$$Md = 160,18$$

✓ Média aritmética, moda e mediana com distribuição de frequência (dados agrupados)

```
[265] dados = {'inferior': [150, 154, 158, 162, 166, 170],  
             'superior': [154, 158, 162, 166, 170, 174],  
             'fi': [5, 9, 11, 7, 5, 3]}
```



```
import pandas as pd  
dataset = pd.DataFrame(dados)  
dataset
```



	inferior	superior	fi
0	150	154	5
1	154	158	9
2	158	162	11
3	162	166	7
4	166	170	5
5	170	174	3



```
[269] dataset['xi'] = (dataset['superior'] + dataset['inferior']) / 2  
dataset
```



	inferior	superior	fi	xi
0	150	154	5	152.0
1	154	158	9	156.0
2	158	162	11	160.0
3	162	166	7	164.0
4	166	170	5	168.0
5	170	174	3	172.0



```
[270] dataset['fi.xi'] = dataset['fi'] * dataset['xi']  
dataset
```



	inferior	superior	fi	xi	fi.xi
0	150	154	5	152.0	760.0
1	154	158	9	156.0	1404.0
2	158	162	11	160.0	1760.0
3	162	166	7	164.0	1148.0
4	166	170	5	168.0	840.0
5	170	174	3	172.0	516.0




```
[271] dataset['Fi'] = 0  
dataset
```



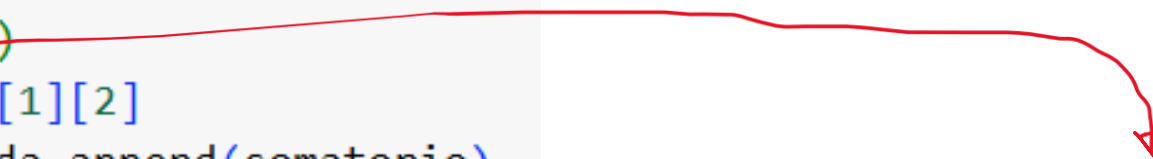
	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	0
1	154	158	9	156.0	1404.0	0
2	158	162	11	160.0	1760.0	0
3	162	166	7	164.0	1148.0	0
4	166	170	5	168.0	840.0	0
5	170	174	3	172.0	516.0	0




```
[272] frequencia_acumulada = []
      somatorio = 0
      for linha in dataset.iterrows():
          #print(linha[1])
          #print(linha[1][2])
          somatorio += linha[1][2]
          frequencia_acumulada.append(somatorio)
```


```
[273] frequencia_acumulada
```

⇒ [5.0, 14.0, 25.0, 32.0, 37.0, 40.0]



	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	0
1	154	158	9	156.0	1404.0	0
2	158	162	11	160.0	1760.0	0
3	162	166	7	164.0	1148.0	0
4	166	170	5	168.0	840.0	0
5	170	174	3	172.0	516.0	0

 dataset['Fi'] = frecuencia_acumulada
dataset



	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	5.0
1	154	158	9	156.0	1404.0	14.0
2	158	162	11	160.0	1760.0	25.0
3	162	166	7	164.0	1148.0	32.0
4	166	170	5	168.0	840.0	37.0
5	170	174	3	172.0	516.0	40.0



▼ Média

```
[275] dataset['fi'].sum(), dataset['fi.xi'].sum()
```

➞ (40, 6428.0)

```
dataset['fi.xi'].sum() / dataset['fi'].sum()
```

➞ 160.7

▼ Moda

```
[277] dataset['fi'].max()
```

➞ 11

```
[278] dataset[dataset['fi'] == dataset['fi'].max()]
```

➞

	inferior	superior	fi	xi	fi.xi	Fi
2	158	162	11	160.0	1760.0	25.0



▶

```
dataset[dataset['fi'] == dataset['fi'].max()][ 'xi'].values[0]
```

➞

160.0

▼ Mediana

dataset							
	inferior	superior	fi	xi	fi.xi	Fi	
0	150	154	5	152.0	760.0	5.0	
1	154	158	9	156.0	1404.0	14.0	
2	158	162	11	160.0	1760.0	25.0	
3	162	166	7	164.0	1148.0	32.0	
4	166	170	5	168.0	840.0	37.0	
5	170	174	3	172.0	516.0	40.0	

```
[281] fi_2 = dataset['fi'].sum() / 2  
      fi_2
```

⇒ 20.0

```
▶ limite_inferior, frequencia_classe, id_frequencia_anterior = 0, 0, 0  
for linha in dataset.iterrows():  
    #print(linha)  
    limite_inferior = linha[1][0]  
    frequencia_classe = linha[1][2]  
    id_frequencia_anterior = linha[0]  
    if linha[1][5] >= fi_2:  
        id_frequencia_anterior -= 1  
        break
```

	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	5.0
1	154	158	9	156.0	1404.0	14.0
2	158	162	11	160.0	1760.0	25.0
3	162	166	7	164.0	1148.0	32.0
4	166	170	5	168.0	840.0	37.0
5	170	174	3	172.0	516.0	40.0

```
[288] limite_inferior, frequencia_classe, id_frequencia_anterior
```

```
➞ (158.0, 11.0, 1)
```

```
[289] Fi_anterior = dataset.iloc[[id_frequencia_anterior]]['Fi'].values[0]  
      Fi_anterior
```

```
➞ 14.0
```

```
▶ mediana = limite_inferior + ((fi_2 - Fi_anterior) * 4) / frequencia_classe  
  mediana
```

```
➞ 160.1818181818182
```

✓ Função completa

```
[291] def get_estatisticas(dataframe):  
    media = dataset['fi.xi'].sum() / dataset['fi'].sum()  
    moda = dataset[dataset['fi'] == dataset['fi'].max()]['xi'].values[0]  
  
    fi_2 = dataset['fi'].sum() / 2  
    limite_inferior, frequencia_classe, id_frequencia_anterior = 0, 0, 0  
    for i, linha in enumerate(dataset.iterrows()):  
        limite_inferior = linha[1][0]  
        frequencia_classe = linha[1][2]  
        id_frequencia_anterior = linha[0]  
        if linha[1][5] >= fi_2:  
            id_frequencia_anterior -= 1  
            break  
    Fi_anterior = dataset.iloc[[id_frequencia_anterior]]['Fi'].values[0]  
    mediana = limite_inferior + ((fi_2 - Fi_anterior) * 4) / frequencia_classe  
  
    return media, moda, mediana
```



get_estatisticas(dataset)



(160.7, 160.0, 160.1818181818182)

MÉDIA GEOMÉTRICA, HARMÔNICA E QUADRÁTICA



150	151	152	152	153	154	155	155	155	155
156	156	156	157	158	158	160	160	160	160
160	161	161	161	161	162	163	163	164	164
164	165	166	167	168	168	169	170	172	173

$$\bar{g} = \sqrt[n]{x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n}$$

Aplicações na geometria, para comparar lados de prismas

Matemática financeira que envolvem taxa percentual acumulada

$$\bar{h} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}}$$

Avaliar desempenho em aprendizagem de máquina

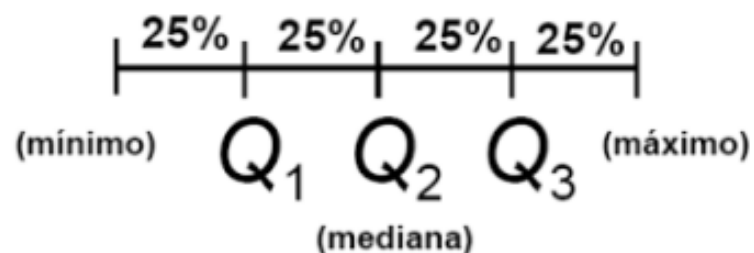
$$QM = \sqrt{\frac{X_1^2 + X_2^2 + \dots + X_n^2}{n}}$$

Aplicações na física

Modelos de regressão

➤ Também é possível implementar essa parte: **média geométrica, harmônica e quadrática** em Python, porém, por ser pouco utilizado não iremos implementar!

QUARTIS



Mediana (ímpar)

$$Mediana = \frac{n}{2}$$

$$Mediana = \frac{9}{2}$$

$$Mediana = 4,5$$

$$Mediana = 5 \text{ (arredondado)}$$

Mediana (par)

$$m = \frac{n}{2}$$

$$m = 2$$

$$m = \frac{151 + 152}{2}$$

$$m = 151,5$$

Mediana (par)

$$m = \frac{n}{2}$$

$$m = 2$$

$$m = \frac{155 + 155}{2}$$

$$m = 155$$

✓ Quartis

```
[50] dados_impar = [150, 151, 152, 152, 153, 154, 155, 155, 155]
```

✓ Cálculo manual

```
[51] np.median(dados_impar)
```

```
↔ 153.0
```

```
[52] posicao_mediana = math.floor(len(dados_impar) / 2)
posicao_mediana
```

```
↔ 4
```

```
[53] esquerda = dados_impar[0:posicao_mediana]
esquerda
```

```
↔ [150, 151, 152, 152]
```

```
▶ np.median(esquerda)
```

```
↔ 151.5
```

```
[55] direita = dados_impar[posicao_mediana + 1:]
direita
```

```
↔ [154, 155, 155, 155]
```

```
▶ np.median(direita)
```

```
↔ 155.0
```

▼ Bibliotecas

▼ numpy

```
[57] np.quantile(dados_impar, 0.5)
```

```
⇒ 153.0
```

```
[58] np.quantile(dados_impar, 0.75)
```

```
⇒ 155.0
```

```
[59] np.quantile(dados_impar, 0.25)
```

```
⇒ 152.0
```

```
esquerda2 = dados_impar[0:posicao_mediana + 1]  
esquerda2
```

```
[150, 151, 152, 152, 153]
```

```
np.median(esquerda2)
```

```
152.0
```

```
np.quantile(dados, 0.25), np.quantile(dados, 0.50), np.quantile(dados, 0.75)
```

```
(155.75, 160.0, 164.0)
```

▼ scipy

```
[1]: stats.scoreatpercentile(dados, 25), stats.scoreatpercentile(dados, 50), stats.scoreatpercentile(dados, 75)
```

⇒ (155.75, 160.0, 164.0)

▼ pandas

```
[1]: import pandas as pd  
dataset = pd.DataFrame(dados)  
dataset.head()
```

⇒

	0
0	150
1	151
2	152
3	152
4	153

[▶] dataset.quantile([0.25, 0.5, 0.75])



0



0.25 155.75



0.50 160.00

0.75 164.00



dataset.describe()



0



count 40.000000



mean 160.375000

std 5.903877

min 150.000000

25% 155.750000

50% 160.000000

75% 164.000000

max 173.000000

QUARTIS – DADOS AGRUPADOS

Estatura (cm)	f_i	x_i	$f_i \cdot x_i$	F_i
150 -- 154	5	152	760	5
154 -- 158	9	156	1404	14
158 -- 162	11	160	1760	25
162 -- 166	7	164	1148	32
166 -- 170	5	168	840	37
170 -- 174	3	172	516	40
Total	40		6428	

$$\frac{\sum f_i}{4} = \frac{40}{4} = 10$$

$$Q1 = l + \frac{(\frac{\sum f_i}{4} - F_{ant}).h}{f_i}$$

$$Q1 = 154 + \frac{(10 - 5).4}{9}$$

$$Q1 = 156,22$$

$$\frac{3 \sum f_i}{4} = \frac{120}{4} = 30$$

$$Q3 = l + \frac{(\frac{3 \sum f_i}{4} - F_{ant}).h}{f_i}$$

$$Q3 = 162 + \frac{(30 - 25).4}{7}$$

$$Q3 = 164,85$$

✓ Quartis com distribuição de frequência (dados agrupados)

[99] dataset



	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	5.0
1	154	158	9	156.0	1404.0	14.0
2	158	162	11	160.0	1760.0	25.0
3	162	166	7	164.0	1148.0	32.0
4	166	170	5	168.0	840.0	37.0
5	170	174	3	172.0	516.0	40.0





```
def get_quartil(dataframe, q1 = True):  
    if q1 == True:  
        fi_4 = dataset['fi'].sum() / 4  
    else:  
        fi_4 = (3 * dataset['fi'].sum()) / 4  
  
    limite_inferior, frequencia_classe, id_frequencia_anterior = 0, 0, 0  
    for linha in dataset.iterrows():  
        limite_inferior = linha[1][0]  
        frequencia_classe = linha[1][2]  
        id_frequencia_anterior = linha[0]  
        if linha[1][5] >= fi_4:  
            id_frequencia_anterior -= 1  
            break  
    Fi_anterior = dataset.iloc[[id_frequencia_anterior]]['Fi'].values[0]  
    q = limite_inferior + ((fi_4 - Fi_anterior) * 4) / frequencia_classe  
  
    return q
```

```
[101] get_quartil(dados), get_quartil(dados, q1 = False)
```

```
→ (156.22222222222223, 164.85714285714286)
```

Q1

Q3

▼ Percentis

```
[107] np.median(dados)
```

```
⇒ 160.0
```

```
[108] np.quantile(dados, 0.5)
```

```
⇒ 160.0
```

```
[109] np.percentile(dados, 50)
```

```
⇒ 160.0
```

```
[110] np.percentile(dados, 5), np.percentile(dados, 10), np.percentile(dados, 90)
```

```
⇒ (151.95, 152.9, 168.1)
```

```
▶ stats.scoreatpercentile(dados, 5), stats.scoreatpercentile(dados, 10), stats.scoreatpercentile(dados, 90)
```

```
⇒ (151.95000000000002, 152.89999999999998, 168.1)
```

```
[113] import pandas as pd  
dataset = pd.DataFrame(dados)  
dataset.head()
```



0



0 150



1 151

2 152

3 152

4 153

```
[115] dataset.quantile([0.05, 0.10, 0.90])
```



0



0.05 151.95



0.10 152.90


0.90 168.10

EXERCÍCIO 01

- O objetivo desta tarefa é gerar estatísticas para o atributo **age** da base de dados do censo.
- Carregue ao arquivo census.csv
- Calcule a média aritmética, média harmônica, média geométrica, [REDACTED] a mediana e a moda.
- Compare os resultados.

Exercício

✓
0s [117] dataset = pd.read_csv('census.csv')

✓
0s  dataset.head()



	age	workclass	final-weight	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black

```
[119] dataset['age'].mean()
```

```
↔ 38.58164675532078
```

```
[120] stats.hmean(dataset['age'])
```

```
↔ 33.91874139089839
```

```
[121] from scipy.stats.mstats import gmean  
      gmean(dataset['age'])
```

```
↔ 36.210879158177256
```

```
[123] dataset['age'].median()
```

```
↔ 37.0
```

```
▶ statistics.mode(dataset['age'])
```

```
↔ 36
```

AMPLITUDE TOTAL E DIFERENÇA INTERQUARTIL

150	151	152	152	153	154	155	155	155
	151,5			Mediana			155	
	Q1			Q2			Q3	

Amplitude total (AT)

$$AT = X_{(\max)} - X_{(\min)} = 155 - 150 = 5$$

$$\text{Cerca inferior} = Q1 - (1.5 * DI)$$

$$\text{Cerca superior} = Q3 + (1.5 * DI)$$

Diferença interquartil

$$Q3 - Q1 = 155 - 151,5 = 3,5$$

Outliers

$$\text{Cerca inferior} = Q1 - (1.5 * DI) = 146,25$$

$$\text{Cerca superior} = Q3 + (1.5 * DI) = 160,25$$

▼ Amplitude total e diferença interquartil

[125] dados

```
array([150, 151, 152, 152, 153, 154, 155, 155, 155, 155, 156, 156, 156,  
       157, 158, 158, 160, 160, 160, 160, 160, 161, 161, 161, 161, 162,  
       163, 163, 164, 164, 164, 165, 166, 167, 168, 168, 169, 170, 172,  
       173])
```

[126] dados.max() - dados.min()

```
23
```

```
[127] q1 = np.quantile(dados, 0.25)  
       q3 = np.quantile(dados, 0.75)  
       q1, q3
```

```
(155.75, 164.0)
```

```
diferenca_interquartil = q3 - q1  
diferenca_interquartil
```

```
8.25
```

```
[129] inferior = q1 - (1.5 * diferenca_interquartil)  
       inferior
```

```
143.375
```

```
superior = q3 + (1.5 * diferenca_interquartil)  
superior
```

```
176.375
```

VARIÂNCIA, DESVIO PADRÃO E COEFICIENTE DE VARIAÇÃO

150	151	152	152	153	154	155	155	155
-----	-----	-----	-----	-----	-----	-----	-----	-----

$$\bar{X} = \frac{\sum x_i}{n} = \frac{150 + 151 + 152 + 152 + 153 + 154 + 155 + 155 + 155}{9} = 153$$

Desvio

$$\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{N}$$
$$\begin{aligned} \text{Desvio} &= 3 \ 2 \ 1 \ 1 \ 0 \ 1 \ 2 \ 2 \ 2 \\ &3^2 + 2^2 + 1^2 + 1^2 + 0^2 + 1^2 + 2^2 + 2^2 + 2^2 \\ &9 + 4 + 1 + 1 + 0 + 1 + 4 + 4 + 4 \\ &28 / 9 = 3,11 \end{aligned}$$

$$\text{Desvio padrão} = \sqrt{3,11} = 1,76$$

“Erro” se substituirmos pelo valor da média

$$CV = \frac{\sigma}{\bar{X}} \cdot 100$$

$$CV = \frac{1,76}{153} \cdot 100 = 1,15\%$$

$$2^2 = 4$$

$$10^2 = 100$$



o quão longe os valores estão do “valor esperado”

▼ Variância, desvio padrão e coeficiente de variação

```
[131] dados_impar = np.array([150, 151, 152, 152, 153, 154, 155, 155, 155])
```

▼ Cálculo manual

```
[132] media = dados_impar.sum() / len(dados_impar)
      media
```

```
⇒ 153.0
```

```
[133] desvio = abs(dados_impar - media)
      desvio
```

```
⇒ array([3., 2., 1., 1., 0., 1., 2., 2., 2.])
```

```
▶ desvio = desvio ** 2
  desvio
```

```
⇒ array([9., 4., 1., 1., 0., 1., 4., 4., 4.])
```

```
[146] soma_desvio = desvio.sum()  
      soma_desvio
```

↩ 28.0

```
[147] v = soma_desvio / len(dados_impar)  
      v
```

↩ 3.1111111111111111

```
[148] dp = math.sqrt(v)  
      dp
```

↩ 1.7638342073763937

▶

```
cv = (dp / media) * 100  
cv
```

↩ 1.1528328152786886

```
[150] def get_variancia_desvio_padrao_coeficiente(dataset):  
      media = dataset.sum() / len(dataset)  
      desvio = abs(dados_impar - media)  
      desvio = desvio ** 2  
      soma_desvio = desvio.sum()  
      variancia = soma_desvio / len(dados_impar)  
      dp = math.sqrt(variancia)  
      return variancia, dp, (dp / media) * 100
```

▶

```
get_variancia_desvio_padrao_coeficiente(dados_impar)
```

↩ (3.1111111111111111, 1.7638342073763937, 1.1528328152786886)

▼ Bibliotecas

```
[152] np.var(dados_impar)
```

```
↔ 3.1111111111111111
```

```
[153] np.std(dados_impar)
```

```
↔ 1.7638342073763937
```

```
[154] np.var(dados)
```

```
↔ 33.984375
```

```
[155] np.std(dados)
```

```
↔ 5.829611908180509
```

```
[156] statistics.variance(dados)
```

```
↔ 34
```

```
[158] statistics.stdev(dados)
```

```
↔ 5.830951894845301
```

```
[159] from scipy import ndimage  
      ndimage.variance(dados)
```

```
↔ 33.984375
```

```
[160] stats.tstd(dados, ddof = 0)
```

```
↔ 5.829611908180509
```

```
[161] stats.variation(dados_impar) * 100
```

```
↔ 1.1528328152786886
```

```
▶ stats.variation(dados) * 100
```

```
↔ 3.634987939629312
```

DESVIO PADRÃO – DADOS AGRUPADOS

Estatura (cm)	f_i	x_i	$f_i \cdot x_i$	x_i^2	$f_i \cdot x_i^2$	F_i
150 -- 154	5	152	760	23104	115520	5
154 -- 158	9	156	1404	24336	219024	14
158 -- 162	11	160	1760	25600	281600	25
162 -- 166	7	164	1148	26896	188272	32
166 -- 170	5	168	840	28224	141120	37
170 -- 174	3	172	516	29584	88752	40
Total	40		6428		1034288	



$$\bar{x} = 160,7$$



$$dp = \sqrt{\frac{\sum f_i \cdot x_i^2}{\sum f_i} - \left(\frac{\sum f_i \cdot x_i}{\sum f_i}\right)^2}$$

$$dp = \sqrt{\frac{1034288}{40} - \left(\frac{6428}{40}\right)^2}$$

$$dp = \sqrt{25857,2 - (160,7)^2}$$

$$dp = \sqrt{25857,2 - (25824,49)}$$

$$dp = 5,71$$

✓ Desvio padrão com dados agrupados

[172] dataset

	inferior	superior	fi	xi	fi.xi	Fi
0	150	154	5	152.0	760.0	5.0
1	154	158	9	156.0	1404.0	14.0
2	158	162	11	160.0	1760.0	25.0
3	162	166	7	164.0	1148.0	32.0
4	166	170	5	168.0	840.0	37.0
5	170	174	3	172.0	516.0	40.0

```
[173] dataset['xi_2'] = dataset['xi'] * dataset['xi']  
dataset
```

	inferior	superior	fi	xi	fi.xi	Fi	xi_2
0	150	154	5	152.0	760.0	5.0	23104.0
1	154	158	9	156.0	1404.0	14.0	24336.0
2	158	162	11	160.0	1760.0	25.0	25600.0
3	162	166	7	164.0	1148.0	32.0	26896.0
4	166	170	5	168.0	840.0	37.0	28224.0
5	170	174	3	172.0	516.0	40.0	29584.0


```
[174] dataset['fi_xi_2'] = dataset['fi'] * dataset['xi_2']  
dataset
```



	inferior	superior	fi	xi	fi.xi	Fi	xi_2	fi_xi_2
0	150	154	5	152.0	760.0	5.0	23104.0	115520.0
1	154	158	9	156.0	1404.0	14.0	24336.0	219024.0
2	158	162	11	160.0	1760.0	25.0	25600.0	281600.0
3	162	166	7	164.0	1148.0	32.0	26896.0	188272.0
4	166	170	5	168.0	840.0	37.0	28224.0	141120.0
5	170	174	3	172.0	516.0	40.0	29584.0	88752.0





```
dataset.columns
```



```
Index(['inferior', 'superior', 'fi', 'xi', 'fi.xi', 'xi_2', 'fi_xi_2', 'Fi'], dtype='object')
```

```
[176] columnas_ordenadas = ['inferior', 'superior', 'fi', 'xi', 'fi.xi', 'xi_2', 'fi_xi_2', 'Fi']
```

```
[177] dataset = dataset[columnas_ordenadas]  
dataset
```



	inferior	superior	fi	xi	fi.xi	xi_2	fi_xi_2	Fi
0	150	154	5	152.0	760.0	23104.0	115520.0	5.0
1	154	158	9	156.0	1404.0	24336.0	219024.0	14.0
2	158	162	11	160.0	1760.0	25600.0	281600.0	25.0
3	162	166	7	164.0	1148.0	26896.0	188272.0	32.0
4	166	170	5	168.0	840.0	28224.0	141120.0	37.0
5	170	174	3	172.0	516.0	29584.0	88752.0	40.0



```
[178] dp = math.sqrt(dataset['fi_xi_2'].sum() / dataset['fi'].sum() - math.pow(dataset['fi.xi'].sum() / dataset['fi'].sum(), 2))  
dp
```



```
5.719265687131764
```

EXERCÍCIO 02

- O objetivo deste exercício é verificar como você pode trabalhar com valores faltantes utilizando as medidas de posição: média.
- Carregue o arquivo **credit_data.csv**
- Vamos identificar os valores faltantes e substituir estes pelo valor da média.

Valores faltantes com média e moda

Média

```
[181] import pandas as pd  
      dataset = pd.read_csv('credit_data.csv')
```

```
[183] dataset.isnull().sum()
```

```
i#clientid    0  
income        0  
age           3  
loan          0  
c#default     0  
dtype: int64
```

```
nulos = dataset[dataset.isnull().any(axis=1)]  
nulos
```

	i#clientid	income	age	loan	c#default
28	29	59417.805406	NaN	2082.625938	0
30	31	48528.852796	NaN	6155.784670	0
31	32	23526.302555	NaN	2862.010139	0

Próximas etapas:

[Gerar código com nulos](#)[Ver gráficos recomendados](#)

```
[186] dataset['age'].mean(), dataset['age'].median()
```

```
(40.80755937840458, 41.3171591130085)
```

```
[187] dataset['age'] = dataset['age'].replace(to_replace = np.nan, value =
```

```
dataset[dataset.isnull().any(axis=1)]
```

```
i#clientid income age loan c#default
```



EXERCÍCIO 03

- O objetivo deste exercício é verificar como você pode trabalhar com valores faltantes utilizando as medidas de posição: moda.
- Carregue o arquivo **autos.csv**
- Vamos identificar os valores faltantes e substituir estes pelo valor da moda.

```
[195] dataset = pd.read_csv('autos.csv', encoding='ISO-8859-1')
```

dataset.head()

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0	golf	150000	0
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	NaN	125000	5
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	8
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	6
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69	fabia	90000	7



```
dataset.isnull().sum()
```



dateCrawled	0
name	0
seller	0
offerType	0
price	0
abtest	0
vehicleType	37869
yearOfRegistration	0
gearbox	20209
powerPS	0
model	20484
kilometer	0
monthOfRegistration	0
fuelType	33386
brand	0
notRepairedDamage	72060
dateCreated	0
nrOfPictures	0
postalCode	0
lastSeen	0
dtype: int64	

```
[225] dataset['fuelType'].unique()
```

```
⇒ array(['benzin', 'diesel', nan, 'lpg', 'andere', 'hybrid', 'cng',  
        'elektro'], dtype=object)
```

```
[226] statistics.mode(dataset['fuelType'])
```

```
⇒ 'benzin'
```

```
[227] dataset['fuelType'] = dataset['fuelType'].replace(to_replace = np.nan, value = statistics.mode(dataset['fuelType']))
```

```
▶ dataset['fuelType'].unique()
```

```
⇒ array(['benzin', 'diesel', 'lpg', 'andere', 'hybrid', 'cng', 'elektro'],  
        dtype=object)
```