

## Aula 2 – Estatística – Resumo – Parte prática

Definir o conjunto de dados:

Pd.series() = conjunto de dados

```
import pandas as pd
# Conjunto de dados
data = pd.Series([128, 100, 180, 150, 200, 90, 340, 105, 85, 270,
                  200, 65, 230, 150, 150, 120, 130, 80, 230, 200,
                  110, 126, 170, 132, 140, 112, 90, 340, 170, 190])
print(data)
```

Data.min() = pegar o valor mínimo do conjunto 'data'

Data.max() = pegar o valor máximo do conjunto 'data'

### Cálculo da frequência ocorrida de dados em cada classe:

Nome\_tabela = Value\_counts(bins=numerodeclasses)

Sort\_index() = ordena índices(classes) em ordem crescente

```
dist_freq = data.value_counts(bins=7).sort_index()
print(dist_freq)
```

```
(64.72399999999999, 104.286]    6
(104.286, 143.571]             9
(143.571, 182.857]             6
(182.857, 222.143]             4
(222.143, 261.429]             2
(261.429, 300.714]             1
(300.714, 340.0]               2
Name: count, dtype: int64
```

Reconfiguração do conjunto de dados:

Pd.DataFrame(nome da tabela definida no series) = criar uma tabela no pandas

Nome\_da\_tabela.reset\_index(nome da tabela) = resetar índices

**Pegar uma coluna na tabela = nome\_da\_tabela['nome da coluna']**

Cumsum() = frequência acumulada

```
# Cálculo da amplitude de classe, com respectivos intervalos de classe
amplitude = round((v_max-v_min)/n_classes)
intervalos = pd.interval_range(start=v_min, end=v_max, freq=amplitude)
print(intervalos)
```

```
# Inserção da coluna Pontos médios
nova_dist_freq['Pontos médios'] = pts_medios
display(nova_dist_freq)
```

```
# Cálculo da amplitude de classe, com respectivos intervalos de classe
amplitude = round((v_max-v_min)/n_classes)
intervalos = pd.interval_range(start=v_min, end=v_max, freq=amplitude)
print(intervalos)
```

```
# Cálculo dos pontos médios das classe
pts_medios = [inter.mid for inter in intervalos]
print(pts_medios)
```

```
[84.5, 123.5, 162.5, 201.5, 240.5, 279.5, 318.5]
```

1. `pts_medios = [inter.mid for inter in intervalos]`: Esta linha cria uma lista chamada `pts_medios` por meio de uma list comprehension. A lista será preenchida com os pontos médios das classes. A estrutura da list comprehension é a seguinte:
  - `inter.mid` é a expressão que calcula o ponto médio de cada classe. Parece que `inter` é uma variável que representa os intervalos das classes.
  - `for inter in intervalos` é um loop que itera por cada intervalo de classe contido na lista `intervalos`. Isso significa que o código calculará o ponto médio de cada intervalo de classe na lista.

```
# Inserção da coluna Pontos médios
nova_dist_freq['Pontos médios'] = pts_medios
display(nova_dist_freq)
```

	Classe	Frequência	Pontos médios
0	(65, 104]	6	84.5
1	(104, 143]	9	123.5
2	(143, 182]	6	162.5
3	(182, 221]	4	201.5
4	(221, 260]	2	240.5
5	(260, 299]	1	279.5
6	(299, 338]	2	318.5

```
# Cálculo e inserção das colunas de frequência relativa e acumulada
observacoes = len(data)
nova_dist_freq['Frequência relativa'] = nova_dist_freq
['Frequência']/observacoes
nova_dist_freq['Frequência acumulada'] = nova_dist_freq
['Frequência'].cumsum()
display(nova_dist_freq)
```

Python

	Classe	Frequência	Pontos médios	Frequência relativa	Frequência acumulada
0	(65, 104]	6	84.5	0.200000	6
1	(104, 143]	9	123.5	0.300000	15
2	(143, 182]	6	162.5	0.200000	21
3	(182, 221]	4	201.5	0.133333	25
4	(221, 260]	2	240.5	0.066667	27
5	(260, 299]	1	279.5	0.033333	28
6	(299, 338]	2	318.5	0.066667	30

**Len()** = calcula a quantidade de elementos na variável 'data'.

**Freq relativa** = frequência da classe/tamanho da amostra(amostra=soma das frequências f).

**Freq acumulada** = soma das frequências dessa classe com todas as anteriores.

## Histograma de frequências

Histograma é construído usando o método **hist** do pandas

A biblioteca matplotlib é utilizada para criar gráficos

```
import matplotlib.pyplot as plt

# Construção do histograma
histograma = data.hist(bins=[inter.left for inter in intervalos]+[v_max],
                        color='blue', edgecolor='black', grid=False)
# Inserção de atributos ao gráfico
histograma.set(xlabel='Preço [US$]',
               ylabel='Frequência - Número de navegadores GPS',
               title='Distribuição de frequências dos preços de GPS',
               xticks=nova_dist_freq['Pontos médios'],
               yticks=range(0, nova_dist_freq['Frequência'].max()+2, 2))

plt.show()
```

**(bins=[inter.left for inter in intervalos]+[v\_max], color='blue', edgecolor='black', grid=False):** Este código contém o bins, que neste caso, tem a função de pegar o limite mínimo (inter.left - primeiro número da coluna 'classes') e o limite máximo (v\_max - maior número da coluna classes), e criar um parâmetro de quantas colunas criar a partir desses limites. A cor das barras é azul, o contorno é preto e não tem grade no fundo do gráfico.

**Histograma = data.hist:** este primeiro código é responsável por criar um histograma ('hist') da tabela 'data'.

**Bins:** Define o número total de barras no gráfico. Ele também pode ser responsável por coletar um limite mínimo e um limite máximo para ter um parâmetro de quantas barras podem ser criadas.

**inter.left for inter in intervalos:** limite da esquerda da variável 'intervalos' (O primeiro número da coluna 'classes').

**Grid:** grade no fundo do gráfico (true/false)

**Histograma.set:** definir algo na variável 'histograma'.

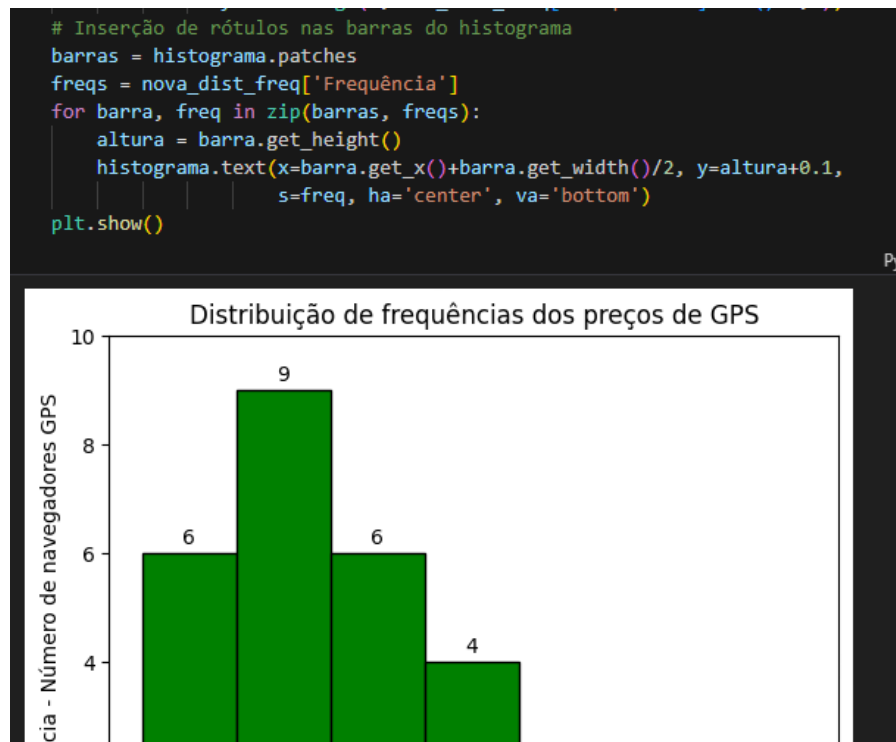
**Xlabel:** 'Legenda horizontal do histograma'

**Ylabel:** 'Legenda vertical do histograma'

**xticks=nova\_dist\_freq['Pontos médios']:** Os números que estão no lado horizontal do histograma são coletados da coluna 'Pontos médios' da tabela 'nova\_dist\_freq'.

**yticks=range(0, nova\_dist\_freq['Frequência'].max()+2, 2):** Os números que estão no lado vertical do histograma são calculados da seguinte forma: o início/primeiro número começa do zero, e vai de 2 em 2 até 2 números maior que o maior número da coluna 'frequência'.

Inserir frequencia no topo de cada barra:



Patches = configuração para modelar as barras

Zip = pegar dois conjuntos e transformar em um só (1º valor de uma lista com o 1º de outra e junta em um só e assim, sucessivamente)

altura = barra.get\_height(): Coleta a altura da barra