Estatística Com Python

Parte I – Frequências e Medidas:

Todos os arquivos serão baixados, mas deixo o link do drive de qualquer forma: https://drive.google.com/drive/folders/1MoAf3dLinpNBXVGIIyetaY5qfSwZ31Gt.

1. **Aula 1 – Qual Seu Tipo de Dado:**
   - 1.1. Dependendo do tipo de dados análises diferentes serão feitas. Importante saber quais são os tipos.
     - 1.1.1. Quantitativos: Expressa quantidade de dados. Contagem, mensurações. Ex.:
       - 1.1.1.1. Idade;
       - 1.1.1.2. Renda;
       - 1.1.1.3. Altura.
     - 1.1.2. Qualitativos: Expressa qualidade dos dados. Ex.:
       - 1.1.2.1. UF;
       - 1.1.2.2. Sexo;
       - 1.1.2.3. Cor;
       - 1.1.2.4. Anos de Estudo.
     - 1.1.3. Podemos dividir os qualitativos em ordinais e nominais, e os quantitativos em discretos e contínuos.
   - 1.2. O que aprendemos:
     - 1.2.1. A identificar a versão de uma biblioteca do Python;
     - 1.2.2. A ler um dataset no formato CSV e criar um DataFrame pandas, com o conteúdo deste dataset;
     - 1.2.3. A identificar e classificar as variáveis de um dataset como quantitativas ou qualitativas;
     - 1.2.4. Que as variáveis qualitativas se dividem em ordinais e nominais;
     - 1.2.5. Que as variáveis quantitativas se dividem em discretas e contínuas.

2. **Aula 2 – Distribuição de Frequência:**
   - 2.1. S.value_counts(): conta a quantidade de vezes que uma variável aparece na series. Podemos passar o parâmetro normalize = True, onde ele devolve a porcentagem de cada variável, onde ele soma elas e divide cada um pela soma:

```
dados['Sexo'].value_counts(normalize=True) * 100
```

```
0    69.299844
1    30.700156
```

2.1.1. Podemos usar a porcentagem e a frequência para criar um df de distribuição e frequência qualitativas:

```
dist_freq_qualitativas = pd.DataFrame({'Frequência': frequencia, 'Porcentagem (%)': percentual})
```

	Frequência	Porcentagem (%)
0	53250	69.299844
1	23590	30.700156

2.1.2. Df.rename_axis('nome_index', axis='columns'/'rows', inplace=True): Renomeia a coluna do index:

```
dist_freq_qualitativas.rename(index= {0: 'Masculino', 1: 'Feminino'}, inplace=True)
dist_freq_qualitativas.rename_axis('Sexo', axis='columns', inplace=True)
dist_freq_qualitativas
```

Sexo	Frequência	Porcentagem (%)
Masculino	53250	69.299844
Feminino	23590	30.700156

2.2. Pd.crosstab(variável_linha, variável_coluna): Cria um df onde faz o cruzamento dos dados da linha e da coluna passadas. No caso utilizamos cor e sexo, então ele irá mostrar na tabela quantas pessoas do sexo masculino tiveram de cada cor, bem como os femininos:

```
sexo = {0: 'Masculino',
        1: 'Feminino'}

cor = {0: 'Indígena',
       2: 'Branca',
       4: 'Preta',
       6: 'Amarela',
       8: 'Parda',
       9: 'Sem declaração'}
```

```
frequencia = pd.crosstab(dados.Sexo, dados.Cor)
frequencia.rename(index=sexo, columns=cor, inplace=True)
frequencia
```

Cor / Sexo	Indígena	Branca	Preta	Amarela	Parda
Masculino	256	22194	5502	235	25063
Feminino	101	9621	2889	117	10862

2.2.1. Podemos passar o mesmo parâmetro (normalize=True) que para value_counts() e já receber o valor das porcentagens:

```
frequencia = pd.crosstab(dados.Sexo, dados.Cor, normalize=True) * 1
00
frequencia
```

Cor / Sexo	Indígena	Branca	Preta	Amarela	Parda
Masculino	0.333160	28.883394	7.160333	0.305830	32.617126
Feminino	0.131442	12.520822	3.759761	0.152264	14.135867

2.2.2. Podemos ainda fazer algo bem mais legal. Podemos passar o valor de outra variável do nosso dataset para ser analisado nesses cruzamentos, ou seja, se escolhermos média da renda, por exemplo, ao invés de mostrar quantos homens e mulheres existem de cada cor, ele passa a exibir, em média, quanto os homens e mulheres ganham de renda. Para isso utilizaremos os parâmetros aggfunc=análise_estat' e values = dados['index']:

```
frequencia = pd.crosstab(dados.Sexo, dados.Cor, aggfunc = 'mean', v
alues = dados.Renda)
frequencia
```

Cor / Sexo	Indígena	Branca	Preta	Amarela	Parda
Masculino	1081.710938	2925.744435	1603.861687	4758.251064	1659.577425
Feminino	2464.386139	2109.866750	1134.596400	3027.341880	1176.758516

2.3. Pd.cut(variável, bins=['distribuição'], labels=labels, include_lowest=True/False): criamos uma lista com o intervalo colocado nos bins a partir da variável escolhida fazendo relação com as labels que determinamos, como no caso da renda, onde temos a classificação de A a E

da renda populacional dos nossos dados analisados. Utilizamos essa função para mostrar quais pessoas estão em qual classificação a partir da renda:

```
classes = [0, 1576, 3152, 7880, 15760, 200000]
labels = ['E', 'D', 'C', 'B', 'A']

pd.cut(dados.Renda, bins = classes, labels = labels, include_lowest
=True)
```

```
0        E
1        E
2        E
3        C
4        E
        ..
76835    E
76836    E
76837    E
76838    E
76839    E
Name: Renda, Length: 76840, dtype: category
Categories (5, object): ['E' < 'D' < 'C' < 'B' < 'A']
```

2.3.1. Porém Podemos notar que ele classifica indivíduo por indivíduo. Para saber a frequência podemos colocar o cut dentro do pd.value_counts(), sabendo quantas pessoas estão em cada classificação:

```
frequencia = pd.value_counts(pd.cut(dados.Renda,
                                    bins = classes,
                                    labels = labels,
                                    include_lowest=True))
frequencia
```

```
E    49755
D    16700
C     7599
B     2178
A      608
Name: Renda, dtype: int64
```

2.3.2. Para saber o percentual, basta colocar o normalize=True e multiplicar por 100, como estávamos fazendo com os outros:

```
percentual = pd.value_counts(pd.cut(dados.Renda,
                                    bins = classes,
                                    labels = labels,
                                    include_lowest=True), normalize
=True) * 100
```

```
percentual
```

```
E    64.751432
D    21.733472
C     9.889381
B     2.834461
A     0.791255
Name: Renda, dtype: float64
```

2.3.3. Criamos a tabela de frequência e porcentagem com pd.DataFrame():

```
dist_freq_quantitativas_personalizadas = pd.DataFrame({'Frequência'
: frequencia,
                                                       'Porcentagem
 (%)': percentual})
dist_freq_quantitativas_personalizadas
```

	Frequência	Porcentagem (%)
E	49755	64.751432
D	16700	21.733472
C	7599	9.889381
B	2178	2.834461
A	608	0.791255

2.3.4. E alteramos a ordem do index para crescente, ou seja, de A para baixo com o sorted_index(ascending=True):

```
dist_freq_quantitativas_personalizadas.sort_index(ascending=False)
```

	Frequência	Porcentagem (%)
A	608	0.791255
B	2178	2.834461
C	7599	9.889381
D	16700	21.733472
E	49755	64.751432

2.4. Biblioteca numpy possuí diversas fórmulas matemáticas que podemos usar.

2.4.1. Com isso podemos fazer vários cálculos como log10 dentre vários outros, permitindo com que façamos cálculos como tamanho de classes:

$$k = 1 + \frac{10}{3}\log_{10} n$$

2.4.1.1. N = valor de linhas do nosso df. Basta atribuir o df.shape[0] em n.

2.4.2. Tendo a fórmula nós podemos executar a conta:

```
n = dados.shape[0]
k = 1 + (10/3) * np.log10(n)
k = int(k.round(0))
```

2.4.2.1. Arredondamos porque não tem como fazer 17.26.... classes, apenas números inteiros.

2.4.3. Com isso podemos fazer um cut passando k como as bins para ter essa classe de amplitude fixa. De resto, a criação da frequência é a mesma:

```
frequencia = pd.value_counts(pd.cut(x = dados.Renda, bins = k, incl
ude_lowest = True), sort = False)
frequencia
```

```
(-200.001, 11764.706]       75594
(11764.706, 23529.412]       1022
(23529.412, 35294.118]        169
(35294.118, 47058.824]         19
(47058.824, 58823.529]         16
(58823.529, 70588.235]          5
(70588.235, 82352.941]          4
(82352.941, 94117.647]          1
(94117.647, 105882.353]         6
(105882.353, 117647.059]        0
(117647.059, 129411.765]        1
(129411.765, 141176.471]        0
(141176.471, 152941.176]        0
(152941.176, 164705.882]        0
(164705.882, 176470.588]        0
(176470.588, 188235.294]        0
(188235.294, 200000.0]          3
Name: Renda, dtype: int64
```

2.4.4. O mesmo vale para o percentual:

```
percentual = pd.value_counts(pd.cut(x = dados.Renda, bins = k, incl
ude_lowest = True), sort = False, normalize=True) * 100
percentual
```

```
(-200.001, 11764.706]       98.378449
(11764.706, 23529.412]       1.330036
(23529.412, 35294.118]       0.219938
(35294.118, 47058.824]       0.024727
(47058.824, 58823.529]       0.020822
(58823.529, 70588.235]       0.006507
(70588.235, 82352.941]       0.005206
(82352.941, 94117.647]       0.001301
(94117.647, 105882.353]      0.007808
(105882.353, 117647.059]     0.000000
(117647.059, 129411.765]     0.001301
(129411.765, 141176.471]     0.000000
(141176.471, 152941.176]     0.000000
(152941.176, 164705.882]     0.000000
(164705.882, 176470.588]     0.000000
(176470.588, 188235.294]     0.000000
(188235.294, 200000.0]       0.003904
Name: Renda, dtype: float64
```

```
dist_freq_quantitativas_amplitude_fixa = pd.DataFrame({'Frequência'
: frequencia, 'Porcentagem (%)': percentual})
dist_freq_quantitativas_amplitude_fixa
```

	Frequência	Porcentagem (%)
(-200.001, 11764.706]	75594	98.378449
(11764.706, 23529.412]	1022	1.330036
(23529.412, 35294.118]	169	0.219938
(35294.118, 47058.824]	19	0.024727
(47058.824, 58823.529]	16	0.020822
(58823.529, 70588.235]	5	0.006507
(70588.235, 82352.941]	4	0.005206

2.5. Sns.distplot(dados, kde=bool): cria um histograma com a variável do df passada.

2.5.1. O kde é a exibição de densidade do gráfico. Quando true mostra a linha traçada em cima das colunas no fundo, quando false, só as colunas.

2.5.2. Podemos atribuir esse gráfico a uma variável (ax, por exemplo) para ficar com fácil acesso.

2.5.3. A partir disso podemos configurar o tamanho de exibição do gráfico com ax.set_size_inches(x, y).

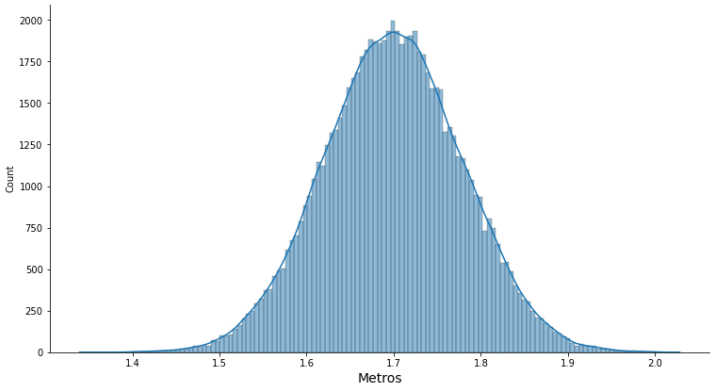
2.5.4. O Título com ax.set_titles('titulo', fontsize=n).

2.5.5. E as labels com ax.set_xlabels('label', fontsize=n) e ax.set_ylabels('label', fontsize=n).

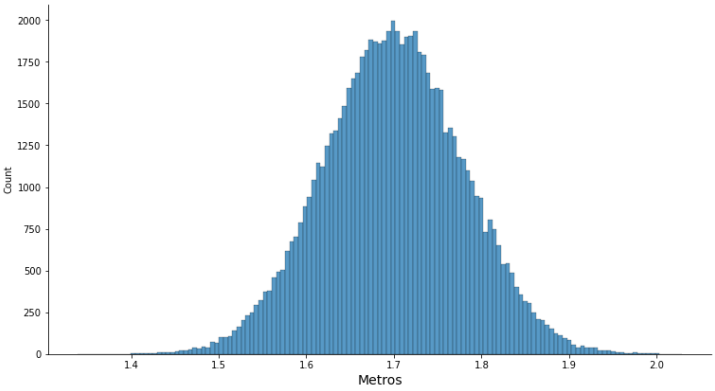
2.5.6. Exemplo:

```
ax = sns.displot(dados.Altura, kde=True)

ax.figure.set_size_inches(12,6)
```

```
ax.set_titles('Distribuição de Frequências - Altura - KDE', fontsiz
e = 18)
ax.set_xlabels("Metros", fontsize = 14)
ax
```

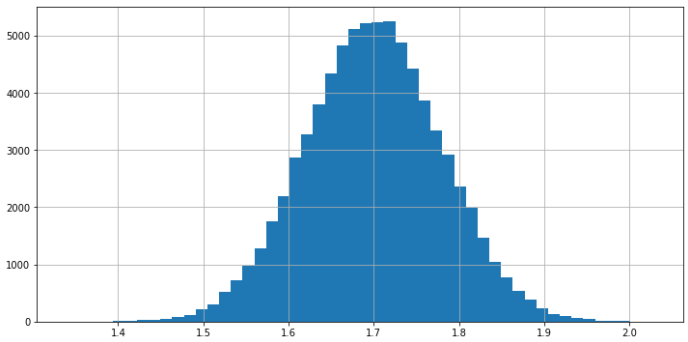


Kde = false:



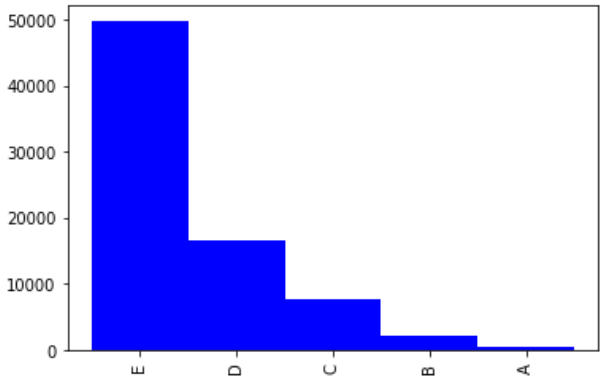
2.5.7. Podemos obter um resultado parecido com o pandas usando a função dados.Altura.hist.(bins=n_barras, figsize=(x,y)):

```
dados.Altura.hist(bins = 50, figsize=(12,6))
```



2.5.8. Com pandas ainda podemos fazer um df[‘variavel’].plot.bar(width=n, color=’color’, alpha=n). Alpah é a transparência da cor das barras:

```
dist_freq_quantitativas_personalizadas.Frequência.plot.bar(width =
1, color = 'blue', alpha = 0.2)
```



2.6. O que aprendemos:

2.6.1. A criar distribuições de frequências (tabelas de frequências) com a função value_counts() do pandas;

2.6.2. A criar distribuições de frequências, com o cruzamento de duas variáveis, utilizando a função crosstab() do pandas;

2.6.3. A criar distribuições de frequências, com classes personalizadas, utilizando as funções value_counts() e cut() conjuntamente;

2.6.4. A utilizar a regra de Sturges para obter um número de classes ótimo para determinado tamanho de amostra;

2.6.5. A plotar o histograma, que é a representação gráfica de uma distribuição de frequências.

3. <u>Aula 3 – Medidas de Tendência Central:</u>

3.1. A média é o ponto de equilíbrio da variável.

3.1.1. As vezes não é o melhor dado para representar um grupo, uma vez que, no caso da renda, se tiver muita gente ganhando pouco e pouca gente ganhando muito, a média não representa essa variável, pois apesar de ser uma média grande, o desvio padrão e a distribuição dos dados são altos, tendo uma baixa representatividade. Cálculo da média:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$$

3.1.2. A média nada mais é do que a soma dos valores dividido pelo seu n total, ou seja:

```
(8 + 10 + 4 + 8 + 6 + 10 + 8) / 7
```

```
7.714285714285714
```

3.1.3. Felizmente não precisamos fazer isso na mão desse jeito sempre, uma vez que seria impossível fazer isso com 70k de dados, por exemplo. Principalmente se forem floats.

3.1.4. Nesse caso o pandas nos ajuda e muito com suas função sum() e seu shape. Com eles podemos pegar apenas uma coluna de informação e passar para o sum que irá somar todos os valores, em seguida podemos fazer a divisão dessa soma pela quantidade de observações do nosso dado, ou seja, quantidade de linhas dele. Para isso usamos o atributo shape[0], que nos retorna exatamente essa informação. No final obtemos o mesmo resultado que feito na mão, assim:

```
df.Fulano.sum() / df.shape[0]
```

```
7.714285714285714
```

3.1.5. Ou então podemos fazer de uma forma ainda MUITO mais fácil: simplesmente pedir a média dessa coluna para o pandas com o mean():

```
df.Fulano.mean()
```

```
7.714285714285714
```

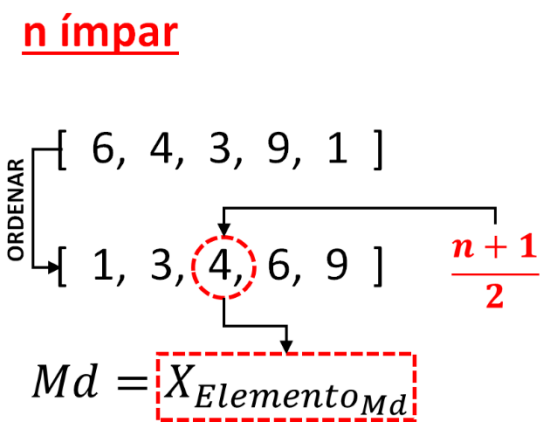
3.1.6. Utilizando esse conhecimento podemos utilizar nossos dados e pegar a média de todas as rendas, ou então usar a função de agrupamento do pandas e fazer a média diretamente a partir de uma variável para esse grupo

df.groupby(‘variável_para_agrupar_por’)[‘variável_analisar’].mean() :

```
dados.groupby(['Sexo'])['Renda'].mean()
```

```
Sexo
0    2192.441596
1    1566.847393
Name: Renda, dtype: float64
```

3.2. Mediana é o valor que divide a series exatamente no meio, por exemplo, a idade de 10 pessoas, organizamos em ordem crescente e vemos qual é a idade que divide exatamente no meio, com os mais novos pra um lado e os mais velhos para o outro.



3.2.1. Cálculos – Identificando Elemento:

3.2.1.1. Quando n for ímpar:

$$Elemento_{Md} = \frac{n+1}{2}$$

3.2.1.2. Quando n for par:

$$Elemento_{Md} = \frac{n}{2}$$

3.2.2. Cálculos – Obtendo a Mediana:

3.2.2.1. Quando n for ímpar:

$$Md = X_{Elemento_{Md}}$$

3.2.2.2. Quando n for par:

$$Md = \frac{X_{Elemento_{Md}} + X_{Elemento_{Md}+1}}{2}$$

3.2.3. Exemplo no python:

```
df = pd.DataFrame(data = {'Fulano': [8, 10, 4, 8, 6, 10, 8],
                          'Beltrano': [10, 2, 0.5, 1, 3, 9.5, 10],
                          'Sicrano': [7.5, 8, 7, 8, 8, 8.5, 7]},
                  index = ['Matemática', 'Português', 'Inglês',
                           'Geografia', 'História', 'Física', 'Quím
ica'])
df.rename_axis('Matérias', axis = 'columns', inplace = True)
df
```

Matérias	Fulano	Beltrano	Sicrano
Matemática	8	10.0	7.5
Português	10	2.0	8.0
Inglês	4	0.5	7.0
Geografia	8	1.0	8.0
História	6	3.0	8.0
Física	10	9.5	8.5
Química	8	10.0	7.0

```
notas_fulano = df.Fulano
notas_fulano = notas_fulano.sort_values()
notas_fulano = notas_fulano.reset_index()
n = notas_fulano.shape[0]
elemento_md = int((n + 1) / 2)
notas_fulano.loc[elemento_md - 1]
```

```
index     Geografia
Fulano            8
Name: 3, dtype: object
```

Ou então:

```
notas_fulano.median()
```

```
Fulano    8.0
dtype: float64
```

3.2.4. Com n par:

```
notas_beltrano = df.Beltrano.sample(6, random_state=101)
notas_beltrano
```

```
Matemática    10.0
Inglês         0.5
Física         9.5
História       3.0
Química       10.0
Português      2.0
Name: Beltrano, dtype: float64
```

```
notas_beltrano.median()
```

```
6.25
```

3.3. A moda é sempre o que mais aparece, ou seja, no caso das notas do df acima, temos que a nota modal de fulano é 8, uma vez que ela aparece 3 vezes.

3.3.1. Se tivermos 2 modas chamamos de bimodal.

3.3.2. Mais de 3, multimodal.

3.3.3. Para pegar as modas use df.mode(). Será retornado um df com as modas:

```
df.mode()
```

Matérias	Fulano	Beltrano	Sicrano
0	8	10.0	8.0

3.3.4. Multimodal:

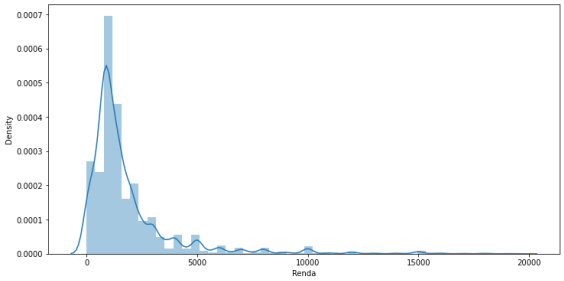
```
exemplo = pd.Series([1, 2, 2, 3, 4, 4, 5, 6, 6])
exemplo.mode()
```

```
0    2
1    4
2    6
dtype: int64
```

3.4. Relação entre média, mediana e moda:

3.4.1. Moda maior que mediana maior que média: Assimetria à direita:

```
ax = sns.distplot(dados.query('Renda < 20000').Renda)
ax.figure.set_size_inches(12,6)
ax
```



3.4.1.1. Tirando a prova nos dados:

```
moda = dados.Renda.mode()[0]
moda
```

```
788
```

```
mediana = dados.Renda.median()
mediana
```

```
1200.0
```

```
media = dados.Renda.mean()
media
```

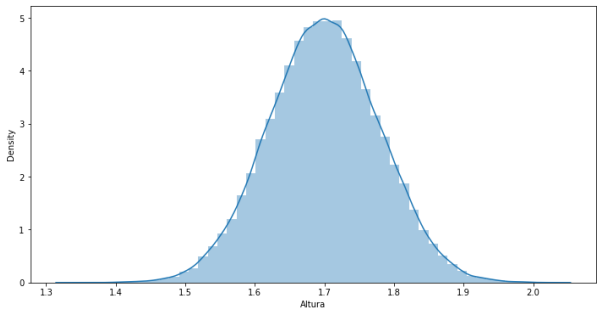
```
2000.3831988547631
```

```
moda < mediana < media
```

```
True
```

3.4.2. Todas iguais (com uma margem de erro): Simétrica:

```
ax = sns.distplot(dados.Altura)
ax.figure.set_size_inches(12,6)
ax
```



3.4.2.1. Prova nos dados:

```
moda = dados.Altura.mode()
moda
```

```
0    1.568128
1    1.671225
2    1.681659
3    1.692977
4    1.708163
5    1.708370
6    1.753842
7    1.779073
8    1.796462
dtype: float64
```

```
mediana = dados.Altura.median()
mediana
```

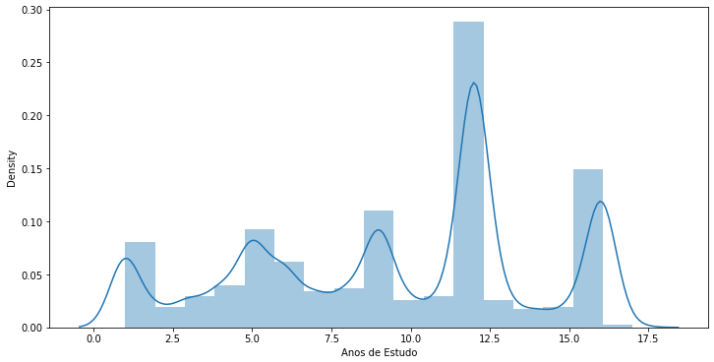
```
1.6993247325
```

```
media = dados.Altura.mean()
media
```

```
1.6995124540575814
```

3.4.3. Média menor que mediana menor que moda: Assimetria à esquerda:

```
ax = sns.distplot(dados['Anos de Estudo'], bins = 17)
ax.figure.set_size_inches(12,6)
ax
```




```
moda = dados['Anos de Estudo'].mode()[0]
moda
```

```
12
```

```
mediana = dados['Anos de Estudo'].median()
mediana
```

```
11.0
```

```
media = dados['Anos de Estudo'].mean()
media
```

```
9.469664237376367
```

```
moda > mediana > media
```

```
True
```

3.5. O que aprendemos:

3.5.1. A calcular as principais medidas de tendência central: média aritmética, mediana e moda;

3.5.2. A identificar características importantes de uma distribuição, como a presença de assimetria e sua direção a partir da relação entre as medidas de tendência central.

4. Aula 4 – Medidas Separatrizes:

4.1. Há uma série de medidas de posição semelhantes na sua concepção à mediana, embora não sejam medidas de tendência central. Como se sabe, a mediana divide a distribuição em duas partes iguais quanto ao número de elementos de cada parte. Já os quartis permitem dividir a distribuição em quatro partes iguais quanto ao número de elementos de cada uma; os decis em dez partes e os centis em cem partes iguais.

4.1.1. S.quantile(.25, .75): Devolve os quartis todos, dependendo do valor passado no parâmetro. Se não passar nada, ele devolve o 0.5, ou seja, a mediana.

4.1.2. Podemos ainda passar uma lista para receber todos os quartis de uma vez:

```
dados.Renda.quantile([0.25, 0.5, 0.75])
```

```
0.25     788.0
0.50    1200.0
0.75    2000.0
Name: Renda, dtype: float64
```

4.1.3. Utilizamos a mesma função para fazer os decis, mas ao invés dos parâmetros que passamos antes, fazemos uma list comprehensions com um for para repetir esse processo 9x:

```
dados.Renda.quantile([i / 10 for i in range(1, 10)])
```

```
0.1     350.0
0.2     788.0
0.3     800.0
0.4    1000.0
0.5    1200.0
0.6    1500.0
0.7    1900.0
0.8    2500.0
0.9    4000.0
```

4.1.3.1. Desse modo temos que 10% das pessoas estão abaixo de 350 enquanto 90 estão acima. O mesmo raciocínio serve para o resto dos decis.

4.1.4. A mesma lógica se aplica aos percentis, mas ao invés de 10 colocamos 100 em tudo:

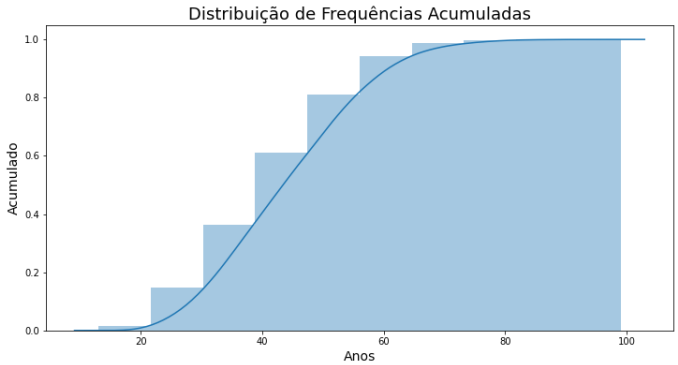
```
dados.Renda.quantile([i / 100 for i in range(1, 100)])
```

```
0.01        0.0
0.02        0.0
0.03        0.0
0.04       50.0
0.05      100.0
           ...
0.95     6000.0
0.96     7000.0
0.97     8000.0
0.98    10000.0
0.99    15000.0
Name: Renda, Leng
```

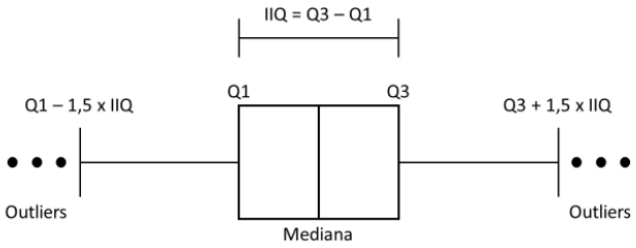
4.1.4.1. A principal diferença é na quantidade de informações que temos, já que ele varia de 1% em 1%. Podemos inclusive notar que 4% das pessoas estão abaixo de 50, mas ao 5% estão abaixo de 100 e assim por diante.

4.1.5. Podemos criar um histograma e visualizar em gráfico o que vimos em números, passando para os parâmetros sns.distplot(dados, hist._kws= {‘cumulative’:True}, kde_kws = {‘cumulative’:True}, bins=10). Ao passar o bins ele vai mostrar 10 barras representando os decis:

```
ax = sns.distplot(dados.Idade, hist_kws= {'cumulative':True},
                  kde_kws = {'cumulative':True}, bins = 10)
ax.figure.set_size_inches(12,6)
ax.set_title('Distribuição de Frequências Acumuladas', fontsize=18)
ax.set_ylabel('Acumulado', fontsize=14)
ax.set_xlabel('Anos', fontsize=14)
ax
```



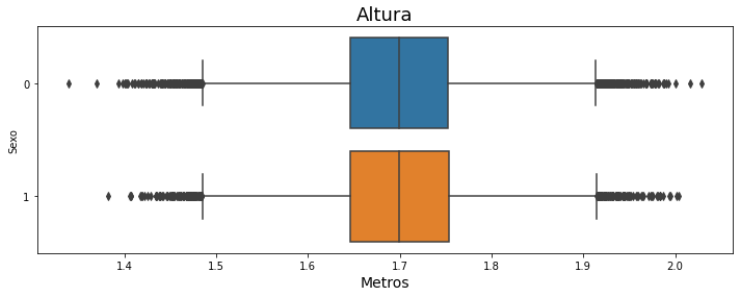
4.2. Box-plot são representações gráficas bem importantes pois mostra muito da distribuição que estamos estudando, como informações de posição, dispersão, simetria, dados discrepantes, dentre outros, e são construídos a partir de medidas separatrizes.



Box-plot

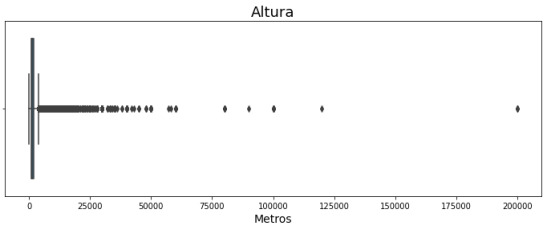
4.2.1. Podemos criar sns.boxplot(x, y~~opt~~, data=data, orient='h') passando mais de 1 parâmetro, como x e y, por exemplo, sendo 2 dados diferentes. Parâmetro orient='h' para deixar os gráficos horizontais:

```
ax = sns.boxplot( x = 'Altura', y = 'Sexo', data = dados, orient =
'h')
ax.figure.set_size_inches(12, 4)
ax.set_title('Altura', fontsize=18)
ax.set_xlabel('Metros', fontsize=14)
ax
```

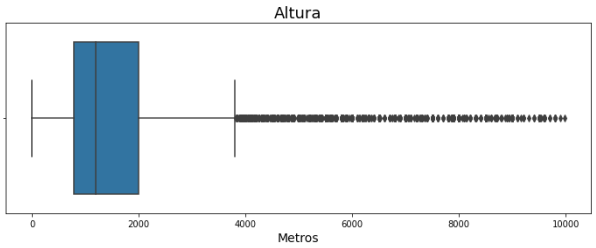


4.2.2. Conseguimos passar uma query para fazer uma seleção de dados, como no caso da renda onde não conseguimos ver nada por causa do outlier:

```
ax = sns.boxplot( x = 'Renda', data = dados, orient = 'h')
ax.figure.set_size_inches(12, 4)
ax.set_title('Altura', fontsize=18)
ax.set_xlabel('Metros', fontsize=14)
ax
```

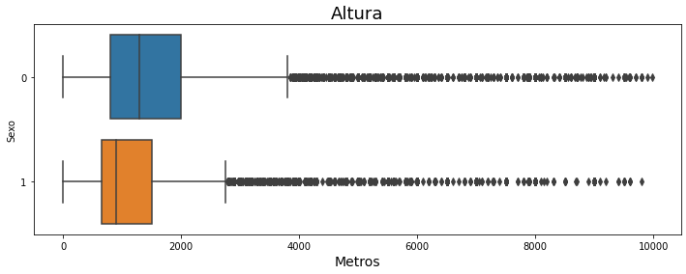


```
ax = sns.boxplot( x = 'Renda', data = dados.query('Renda < 10000'),
 orient = 'h')
ax.figure.set_size_inches(12, 4)
ax.set_title('Altura', fontsize=18)
ax.set_xlabel('Metros', fontsize=14)
ax
```

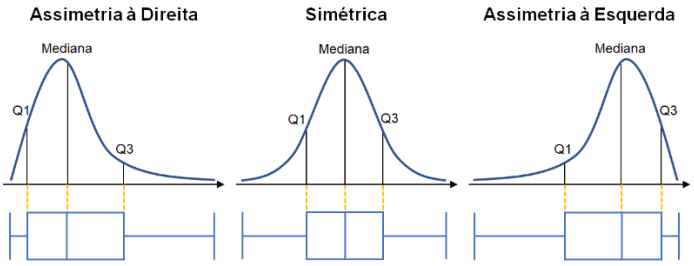


4.2.3. Podemos ainda colocar 2 dados como acima:

```
ax = sns.boxplot( x = 'Renda', y = 'Sexo', data = dados.query('Rend
a < 10000'), orient = 'h')
ax.figure.set_size_inches(12, 4)
ax.set_title('Altura', fontsize=18)
ax.set_xlabel('Metros', fontsize=14)
ax
```



4.2.4. Comparação de simetria:



4.3. O que aprendemos:

4.3.1. A obter os quartis, decis e percentis de uma distribuição;

4.3.2. Também conhecidas como medidas separatrizes, pois dividem uma distribuição em partes iguais;

4.3.3. A construir e interpretar um boxplot, com a utilização dos quartis.

5. Aula 5 – Medidas de Dispersão:

5.1. Embora as medidas de posição forneçam uma sumarização bastante importante dos dados, elas podem não ser suficientes para caracterizar conjuntos distintos, especialmente quando as observações de determinada distribuição apresentarem dados muito dispersos.

5.1.1. Desvio Médio Absoluto: Para fazer esse cálculo podemos fazer na mão como demonstrado abaixo, ou utilizar o s.mad(), o retorno será o mesmo. Cálculo:

$$DM = \frac{1}{n}\sum_{i=1}^{n}|X_i - \bar{X}|$$

5.1.2. Como podemos ver pela fórmula, dm é igual a média do valor absoluto, ou seja, positivo, da somatória dos desvios, sendo este o xi (x índice) menos a média geral:

```
notas_fulano = df[['Fulano']]
nota_media_fulano = notas_fulano.mean()[0]
notas_fulano['Desvio'] = notas_fulano['Fulano'] - nota_media_fulano
notas_fulano['|Desvio|'] = notas_fulano.Desvio.abs()
notas_fulano['|Desvio|'].mean()
```

```
1.5510204081632648
```

5.1.3. Usando o s.mad():

```
desvio_medio_absoluto = notas_fulano.Fulano.mad()
desvio_medio_absoluto
```

```
1.5510204081632648
```

5.1.4. A soma dos desvios em relação a média é 0.

5.1.5. Quando vamos fazer esse cálculo pode ser que tenham números negativos, nesse caso podemos usar o abs() para ignorar o sinal negativo tornando todos positivos:

```
notas_fulano['|Desvio|'] = notas_fulano.Desvio.abs()
notas_fulano
```

Matérias	Fulano	Desvio	|Desvio|
Matemática	8	0.285714	0.285714
Português	10	2.285714	2.285714
Inglês	4	-3.714286	3.714286
Geografia	8	0.285714	0.285714
História	6	-1.714286	1.714286
Física	10	2.285714	2.285714
Química	8	0.285714	0.285714

5.2. Variância: A variância é construída a partir das diferenças entre cada observação e a média dos dados, ou seja, o desvio em torno da média. No cálculo da variância, os desvios em torno da média são elevados ao quadrado.

5.2.1. Variância Populacional: Calcula a variância, ou seja, o desvio em torno da média de uma população toda. Cálculo:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2$$

5.2.2. Variância Amostral: Utiliza-se de um cálculo para ver a variância de uma amostra que representa a população. Cálculo:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

5.2.3. Podemos usar o s.pow(n) do pandas para fazer cálculos de elevados. Substituir n pelo valor a elevar:

```
notas_fulano['(Desvio)^2'] = notas_fulano.Desvio.pow(2)
notas_fulano
```

Matérias	Fulano	Desvio	|Desvio|	(Desvio)^2
Matemática	8	0.285714	0.285714	0.081633
Português	10	2.285714	2.285714	5.224490
Inglês	4	-3.714286	3.714286	13.795918
Geografia	8	0.285714	0.285714	0.081633
História	6	-1.714286	1.714286	2.938776
Física	10	2.285714	2.285714	5.224490
Química	8	0.285714	0.285714	0.081633

5.2.4. Utilizamos isso para fazer a variância, agora basta fazer a somatória geral do desvio^2 e dividir por n – 1:

```
notas_fulano['(Desvio)^2'].sum() / (len(notas_fulano) - 1)
```

```
4.57142857142857
```

5.2.5. Ou então, simplificando e MUITO com o pandas, usamos o s.var():

```
variancia = notas_fulano.Fulano.var()
variancia
```

```
4.57142857142857
```

5.3. Desvio Padrão: Uma das restrições da variância é o fato de fornecer medidas em quadrados das unidades originais - a variância de medidas de comprimento, por exemplo, é em unidades de área. Logo, o fato de as unidades serem diferentes dificulta a comparação da dispersão com as variáveis que a definem. Um modo de eliminar essa dificuldade é considerar sua raiz quadrada.

5.3.1. É a raiz quadrada da variância, ou seja, retira o ^{2} da nossa variância que tem isso como unidade de medida, obtendo uma medida de dispersão média dos dados na mesma unidade de medida do dado que o gerou.

5.3.2. Desvio Padrão Populacional:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2} \Longrightarrow \sigma = \sqrt{\sigma^2}$$

5.3.3. Desvio Padrão Amostral:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2} \Longrightarrow S = \sqrt{S^2}$$

5.3.4. Podemos extrair a raiz quadrada de 2 formas: utilizamos a np.sqrt(variância), ou então a s.std():

```
np.sqrt(variancia)
```

```
2.1380899352993947
```

```
desv_pad = notas_fulano.Fulano.std()
desv_pad
```

```
2.1380899352993947
```

5.4. O que aprendemos:

5.4.1. A obter três medidas de dispersão importantes. O desvio médio absoluto, a variância e o desvio padrão;

5.4.2. Que, embora as medidas de posição forneçam uma sumarização bastante importante dos dados, elas podem não ser suficientes para caracterizar conjuntos distintos, especialmente quando as observações de determinada distribuição apresentarem dados muito dispersos:

5.4.2.1. Por isso, para complementar nossas análises e poder caracterizar melhor o conjunto de dados, utilizamos as medidas de dispersão.

6. Aula 6 – Projeto Final:

6.1. O que aprendemos:

6.1.1. A realizar uma análise descritiva em um conjunto de dados;

6.1.2. Funcionalidades extras da função crosstab() com o parâmetro aggfunc;

6.1.3. A construir boxplots com o cruzamento de três informações distintas.

Parte II – Probabilidade e Amostragem:

1. Aula 1 – Distribuição Binominal:

1.1. Distribuição binomial: Um evento binomial é caracterizado pela possibilidade de ocorrência de apenas duas categorias. Estas categorias somadas representam todo o espaço amostral, sendo também mutuamente excludentes, ou seja, a ocorrência de uma implica na não ocorrência da outra. Em análises estatísticas o uso mais comum da distribuição binomial é na solução de problemas que envolvem situações de sucesso e fracasso. Cálculo:

$$P(k) = \binom{n}{k} p^k q^{n-k}$$

p = probabilidade de sucesso
$q = (1 - p)$ = probabilidade de fracasso
n = número de eventos estudados
k = número de eventos desejados que tenham sucesso

1.1.1. Espaço amostral é a possibilidade de ocorrência de uma probabilidade, como no caso de jogar um dado de 6 lados, a probabilidade de cair 1 deles é 1 em 6.

1.1.2. Requisitos para podermos usar o experimento binomial:

1.1.2.1. Realização de n ensaios idênticos;

1.1.2.2. Os ensaios são independentes;

1.1.2.3. Somente dois resultados são possíveis, exemplo: Verdadeiro ou falso; Cara ou coroa; Sucesso ou fracasso;

1.1.2.4. A probabilidade de sucesso é representada por p e a de fracasso por 1−p=q. Estas probabilidades não se modificam de ensaio para ensaio.

1.1.3. Média da distribuição binomial: O valor esperado ou a média da distribuição binomial é igual ao número de experimentos realizados multiplicado pela chance de ocorrência do evento:

$$\mu = n \times p$$

1.1.4. Desvio padrão da distribuição binomial: O desvio padrão é o produto entre o número de experimentos, a probabilidade de sucesso e a probabilidade de fracasso:

$$\sigma = \sqrt{n \times p \times q}$$

1.1.5. Combinações: Número de combinações de n objetos, tomados k a cada vez, é:

$$C_k^n = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$n! = n \times (n-1) \times (n-2) \times \ldots \times (2) \times (1)$$
$$k! = k \times (k-1) \times (k-2) \times \ldots \times (2) \times (1)$$

Por definição:

$$0! = 1$$

1.1.5.1. Para calcular as combinações, precisamos importar o comb da biblioteca scipy.special:

```
from scipy.special import comb
```

1.1.6. Exemplo de combinações: Em um volante de loteria da Mega Sena temos um total de 60 números para escolher onde a aposta mínima é de seis números. Você que é curiosa(o) resolve calcular a probabilidade de se acertar na Mega Sena com apenas um jogo. Para isso precisamos saber quantas combinações de seis números podem ser formadas com os 60 números disponíveis:

$$C_6^{60} = \binom{60}{6} = \frac{60!}{6!(60-6)!}$$

1.1.7. Resolvendo com o comb do scipy:

```
combinacoes = comb(60, 6)
combinacoes
```

```
50063860.0
```

1.1.7.1. Para calcular a probabilidade de acerto basta pegar o n de números de apostas, por exemplo 1 bilhete de aposta, e dividir pelo resuldado de c^{60}_6, que é 50063860 nesse caso:

```
probabilidade = 1 / combinacoes
probabilidade
```

```
1.9974488583181 56e-08
```
= 0,0000000199744...

1.1.7.2. Podemos fazer uma formatação com o print:

```
print(f'{probabilidade:.15f}')
```

```
0.000000019974489
```

1.2. Quando temos um problema precisamos coletar os itens disponibilizados e ver em que distribuição de probabilidade ele se adequa.

1.2.1. Exemplo: Em um concurso para preencher uma vaga de cientista de dados temos um total de 10 questões de múltipla escolha com 3 alternativas possíveis em cada questão. Cada questão tem o mesmo valor. Suponha que um candidato resolva se aventurar sem ter estudado absolutamente nada. Ele resolve fazer a prova de olhos vendados e chutar todas as respostas. Assumindo que a prova vale 10 pontos e a nota de corte seja 5, obtenha a probabilidade deste candidato acertar 5 questões e também a probabilidade deste candidato passar para a próxima etapa do processo seletivo:

1.2.1.1. Distribuição do tipo binominal: Ao analisar as características colocadas acima para caracterizar essa distribuição, vemos que nosso exemplo se enquadra por ter 10 questões com 3 alternativas e 1 concurso. Cada questão é um experimento diferente com 3 alternativas cada, ou seja, temos n=10. Só existem 2 possibilidades: apesar de 3 alternativas, temos apenas a possibilidade de acertar ou errar.

1.2.1.2. Probabilidade de acertar: p = 1 em 3 ou seja:

```
numero_de_alternativas_por_questao = 3
p = 1 / numero_de_alternativas_por_questao
p
```

```
0.3333333333333333
```
* 100 = 33.333...%

1.2.1.3. Probabilidade de errar: q = 1 - p

```
q = 1 - p
q
```

```
0.6666666666666667
```
* 100 = 66.666...%

1.2.1.4. Total de eventos que se deseja sucesso: k = 5. 5 pois queremos que ele acerte 5 das 10 questões para passar.

1.2.1.5. Solução: Aplicamos a fórmula do começo – a primeira.:

```
probabilidade = (comb(n, k) * (p**k) * (q**(n-k)))
print(f'{probabilidade:.8f}')
```

```
0.13656455
```
* 100 = 13.65%

1.2.1.6. Resolvendo com o python: Usamos a biblioteca binom do scipy.stats, passando binom.pmf(k, n, p) :

```
probabilidade = binom.pmf(k, n, p)
print(f'{probabilidade:.8f}')
```

```
0.13656455
```

1.2.2. Entretanto, apesar de termos feito tudo isso essa foi somente a probabilidade de ele acertar 5, não a dele passar, uma vez que 5 acertos é o mínimo necessário para que ele passe, ele ainda pode acertar 6, 7, 8, 9 e 10 questões:

$$P(acertar \geq 5) = P(5) + P(6) + P(7) + P(8) + P(9) + P10)$$

1.2.2.1. Podemos fazer esse cálculo de várias formas, sendo a primeira delas exatamente igual a figura acima:

```
binom.pmf(5, n, p) + binom.pmf(6, n, p) + binom.pmf(7, n, p) + bino
m.pmf(8, n, p) + binom.pmf(9, n, p) + binom.pmf(10, n, p)
```

```
0.2131280800690952
```
= 21.31%

1.2.2.2. O segundo seria passando ao invés de k, uma lista com todas as outras chances e somando tudo no final:

```
binom.pmf([5, 6, 7, 8, 9, 10], n, p).sum()
```

```
0.2131280800690952
```

1.2.2.3. A terceira é utilizar o binom.cdf(k, n, p), sendo essa função acumulativa, ou seja, ele vai fazer o cálculo e a soma de todas as probabilidades desde 0 até o valor passado em k:

```
binom.cdf(4, n, p)
```

```
0.7868719199309049
```

1.2.2.4. Note que não é o valor que queríamos, mas ao colocar o 1 – binom ele nos retorna exatamente o valor de antes. Isso ocorre por ele ter o caráter acumulativo, então precisamos subtrair 100%, ou seja 1, do valor que ele nos retorna:

```
1 - binom.cdf(4, n, p)
```

```
0.21312808006909512
```

1.2.2.5. Temos ainda o binom.sf(k, n, p) como quarta opção, fazendo exatamente a conta acima: 1 – binom.cdf(k, n, p):

```
binom.sf(4, n, p)
```

```
0.21312808006909512
```

1.2.3. **1-cdf e sf são as probabilidades de acertar 5 questões ou mais. Pmf é a probabilidade de acertar até 5 questões.**

1.3. Aplicação da média:

1.3.1. Exemplo: Uma cidade do interior realiza todos os anos uma gincana para arrecadar fundos para o hospital da cidade. Na última gincana se sabe que a proporção de participantes do sexo feminino foi de 60%. O total de equipes, com 12 integrantes, inscritas na gincana deste ano é de 30. Com as informações acima responda: Quantas equipes deverão ser formadas por 8 mulheres?

1.3.2. Nosso p = 0.6, n = 12 e k = 8:

```
probabilidade = binom.pmf(k, n, p)
print(f'{probabilidade:.8f}')
```

```
0.21284094
```

1.3.3. Nossa pergunta foi quantas equipes poderiam ser formadas por 8 mulheres, para isso fazemos n * p como vimos na fórmula no começo:

```
equipes = 30 * probabilidade
equipes
```

```
6.385228185599988
```

1.3.3.1. Não existe número de equipes quebrado, portanto podemos arredondar, mas se quiser a resposta exata é a acima.

1.4. O que aprendemos:

1.4.1. Os conceitos básicos da distribuição de probabilidades binomial;

1.4.2. A identificar e solucionar problemas que envolvam um experimento binomial;

1.4.3. A obter a probabilidade de eventos que envolvam apenas duas possibilidades de resultado.

2. Aula 2 – Distribuição de Poisson:

2.1. É empregada para descrever o número de ocorrências em um intervalo de tempo ou espaço específico. Os eventos são caracterizados pela possibilidade de contagem dos sucessos, mas a não possibilidade de contagem dos fracassos. Como exemplos de processos onde podemos aplicar a distribuição de Poisson temos a determinação do número de clientes que entram em uma loja em determinada hora, o número de carros que chegam em um drive-thru de uma lanchonete na hora do almoço, a determinação do número de acidentes registrados em um trecho de estrada etc. Cálculo:

$$P(k) = \frac{e^{-\mu}(\mu)^k}{k!}$$

2.1.1. O que é cada elemento:

e = constante cujo valor aproximado é 2,718281828459045

μ = representa o número médio de ocorrências em um determinado intervalo de tempo ou espaço

k = número de sucessos no intervalo desejado

2.1.1.1. O numero de euri, ou seja, o *e* conseguimos obter a partir da numpy.e:

```
import numpy as np
np.e
```

```
2.718281828459045
```

2.1.2. O experimento:

```
1. A probabilidade de uma ocorrência é a mesma em todo o intervalo observado.
2. O número de ocorrências em determinado intervalo é independente do número de ocorrências em outros intervalos.
3. A probabilidade de uma ocorrência é a mesma em intervalos de igual comprimento.
```

2.1.3. A média da distribuição é justamente o u da formula acima:

$$\mu$$

2.1.4. Desvio padrão de Poisson:

$$\sigma = \sqrt{\mu}$$

2.2. Vamos resolver o seguinte exemplo com utilizando essa distribuição:

2.2.1. Um restaurante recebe em média 20 pedidos por hora. Qual a chance de que, em determinada hora escolhida ao acaso, o restaurante receba 15 pedidos?

2.2.2. O próprio exemplo já nos dá no enunciado os elementos da fórmula que precisaremos utilizar:

2.2.2.1. Mi/u/média = 20;

2.2.2.2. K = 15;

2.2.3. Depois disso é só fazer a fórmula:

```
probabilidade = ((np.e ** (-
media)) * (media ** k)) / (np.math.factorial(k))
probabilidade
```

```
0.0516488535317584
```

2.2.3.1. Podemos fazer o fatorial de forma muito simples utilizando a np.math.factoria(k).

2.2.4. Para resolver o mesmo problema de modo muito mais simples, podemos utilizar o poisson.pmf(k, mi/média) da biblioteca scipy.stats:

```
from scipy.stats import poisson
probabilidade = poisson.pmf(k, media)
probabilidade
```

```
0.05164885353175814
```

2.2.5. Ou seja, a probabilidade de um restaurante receber 15 pedidos em 60 minutos, onde já recebe 20 em média, é de 5,16%.

2.3. O que aprendemos:

2.3.1. Os conceitos básicos da distribuição de probabilidades Poisson;

2.3.2. A obter as probabilidades em problemas, como os seguintes:

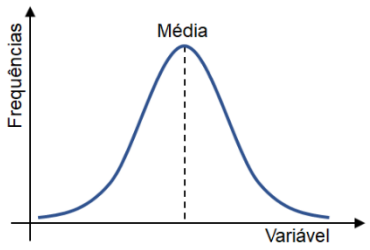
2.3.2.1. Determinação do número de clientes que entram em uma loja em determinada hora;

2.3.2.2. Número de carros que chegam em um drive-thru de uma lanchonete na hora do almoço;

2.3.2.3. Determinação do número de acidentes registrados em um trecho de estrada.

3. Aula 3 – Distribuição Normal:

3.1. A distribuição normal é uma das mais utilizadas em estatística. É uma distribuição contínua, onde a distribuição de frequências de uma variável quantitativa apresenta a forma de sino e é simétrica em relação a sua média.



3.2. Características importantes:

3.2.1. É simétrica em torno da média;

3.2.2. A área sob a curva corresponde à proporção 1 ou 100%;

3.2.3. As medidas de tendência central (média, mediana e moda) apresentam o mesmo valor;

3.2.4. Os extremos da curva tendem ao infinito em ambas as direções e, teoricamente, jamais tocam o eixo x;

3.2.5. O desvio padrão define o achatamento e largura da distribuição. Curvas mais largas e mais achatadas apresentam valores maiores de desvio padrão;

3.2.6. A distribuição é definida por sua média e desvio padrão;

3.2.7. A probabilidade sempre será igual à área sob a curva, delimitada pelos limites inferior e superior.

3.3. Cálculo:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

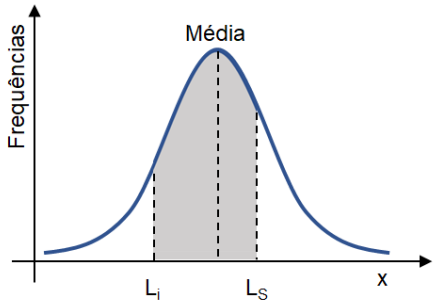
3.3.1. Cada elemento:

x = variável normal

σ = desvio padrão

μ = média

3.3.2. A probabilidade é obtida a partir da área sob a curva, delimitada pelos limites inferior e superior especificados. Um exemplo pode ser visto na figura abaixo.



3.3.3. Para obter a área acima basta calcular a integral da função para os intervalos determinados. Conforme equação abaixo:

$$P(L_i < x < L_s) = \int_{L_i}^{L_s} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

3.3.4. Elementos:

x = variável normal
σ = desvio padrão
μ = média
L_i = limite inferior
L_s = limite superior

3.4. Tabelas padronizadas:

3.4.1. As tabelas padronizadas foram criadas para facilitar a obtenção dos valores das áreas sob a curva normal e eliminar a necessidade de solucionar integrais definidas.

3.4.2. Para consultarmos os valores em uma tabela padronizada basta transformarmos nossa variável em uma variável padronizada Z.

3.4.3. Esta variável Z representa o afastamento em desvios padrões de um valor da variável original em relação à média. Cálculo:

$$Z = \frac{x - \mu}{\sigma}$$

3.4.4. Elementos:

x = variável normal com média μ e desvio padrão σ
σ = desvio padrão
μ = média

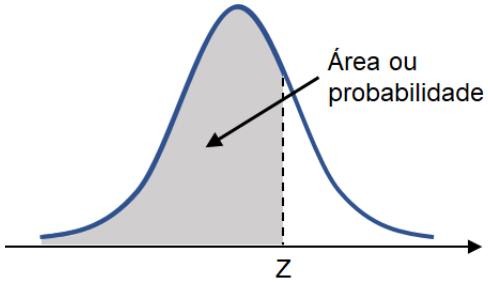
3.4.5. Podemos criar a tabela com o seguinte código, mas é mais fácil consultar o livro:

```
import pandas as pd
import numpy as np
from scipy.stats import norm

tabela_normal_padronizada = pd.DataFrame(
    [],
    index=["{0:0.2f}".format(i / 100) for i in range(0, 400, 10)],
    columns = ["{0:0.2f}".format(i / 100) for i in range(0, 10)])

for index in tabela_normal_padronizada.index:
    for column in tabela_normal_padronizada.columns:
        Z = np.round(float(index) + float(column), 2)
        tabela_normal_padronizada.loc[index, column] = "{0:0.4f}".f
ormat(norm.cdf(Z))
```

```
tabela_normal_padronizada.rename_axis('Z', axis = 'columns', inplac
e = True)

tabela_normal_padronizada
```

Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.00	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.10	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.20	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.30	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.40	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.50	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.60	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.70	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.80	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133

3.4.6. A tabela acima fornece a área sob a curva entre $-\infty$ e Z desvios padrão acima da média. Lembrando que por se tratar de valores padronizados temos $\mu=0$.



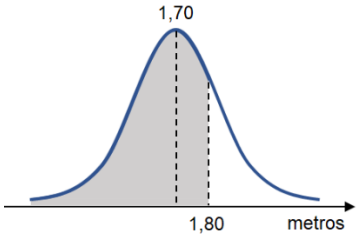
3.5. Exemplo para resolver: Em um estudo sobre as alturas dos moradores de uma cidade verificou-se que o conjunto de dados segue uma distribuição aproximadamente normal, com média 1,70 e desvio padrão de 0,1. Com estas informações obtenha o seguinte conjunto de probabilidades:

3.5.1. Probabilidade de uma pessoa, selecionada ao acaso, ter menos de 1,80 metros.

3.5.2. Probabilidade de uma pessoa, selecionada ao acaso, ter entre 1,60 metros e 1,80 metros.

3.5.3. Probabilidade de uma pessoa, selecionada ao acaso, ter mais de 1,90 metros.

3.6. Primeiro caso:



3.6.1. Calculando Z:

```
media = 1.7
desvio_padrao = 0.1
Z = (1.8 - media) / desvio_padrao
Z
```

```
1.0000000000000009
```

3.6.2. Solução 1: Usando a tabela:

3.6.2.1. Pegamos o valor de Z e consultamos na tabela procurando esse valor, ou seja, se foi 1.00, olhamos na linha 1.00 e na coluna 0.00. Mas se nosso Z tivesse dado 0.15, por exemplo, precisaríamos olhar na linha 0.10 e na coluna 0.05, a intercessão da linha com a coluna seria nossa probabilidade:

Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.00	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.10	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753

3.6.2.2. No nosso caso, com o resultado 1.00 temos que a probabilidade é igual a 0.8413, ou seja, a probabilidade de uma pessoa selecionada ao acaso ter menos de 1.8m é de 84.13%.

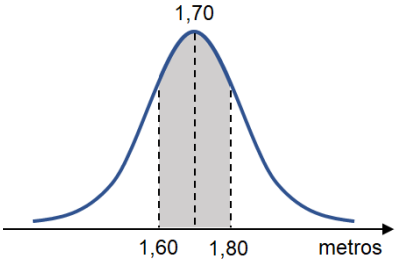
3.6.3. Solução 2: Usando scipy:

```
from scipy.stats import norm
norm.cdf(Z)
```

```
0.8413447460685431
```

3.6.3.1. Utilizando o scipy basicamente nos poupa tempo de procurar o resultado na tabela que é deveras bem grande, mas ainda assim temos que calcular Z.

3.7. Segundo caso:



3.7.1. Solução 1: Tabela

3.7.1.1. Calculamos Z para 1.8 como no caso 1 e o valor da tabela que recebemos pegamos o 0.5, ou seja, 50%, e subtraímos do valor da tabela encontrada pelo Z. Fizemos isso pois queremos saber qual é a área da média, sendo ela 1.7, até 1.8, o resto da curva à direita, que corresponde 50%, ou seja, 0.5, não nos interessa, portanto subtraímos um do outro:

```
Z = (1.8 - media) / desvio_padrao
Z
```

```
1.0000000000000009
```

```
probabilidade = 0.8413 - 0.5
probabilidade
```

```
0.34130000000000005
```

3.7.1.1.1. Com isso temos que nossa área de 1.7 à 1.8 é 0.3413.

3.7.1.2. Como nesse caso também queremos a área correspondente de 1.7 à 1.60 também, e sabemos que equivale ao mesmo valor que 1.7 à 1.8 por ambos variarem 10 da média, podemos só multiplicar a área que descobrimos acima por 2:

```
probabilidade = (0.8413 - 0.5) * 2
probabilidade
```

```
0.6826000000000001
```

3.7.2. Solução 2: Scipy

3.7.2.1. Precisamos calcular o Z inferior e superior para esse cálculo:

```
Z_inferior = (1.6 - media) / desvio_padrao
Z_inferior
```

```
-0.9999999999999987
```

```
Z_superior = (1.8 - media) / desvio_padrao
Z_superior
```

```
1.0000000000000009
```

3.7.2.2. Para calcular a área precisamos subtrair o valor da tabela do limite superior de 1 menos o limite superior novamente:

```
probabilidade = norm.cdf(Z_superior) - (1 - norm.cdf(Z_superior))
probabilidade
```

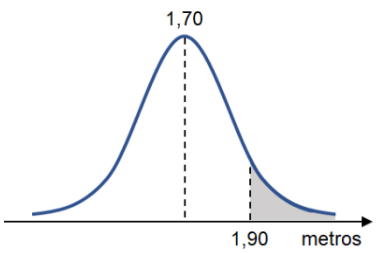
```
0.6826894921370863
```

3.7.2.3. Para deixar ainda mais fácil, podemos simplesmente passar o norm.cdf() do z superior menos o do inferior:

```
probabilidade = norm.cdf(Z_superior) - norm.cdf(Z_inferior)
probabilidade
```

```
0.6826894921370857
```

3.7.3. Solução 3:



3.7.3.1. Quando obtemos o Z e vemos o valor na tabela temos exatamente o valor da área da esquerda para a direita até o ponto que nos é pedido, ou seja, se obtermos o Z de 1.9 e olharmos na tabela, o valor que vamos receber vai ser o total da esquerda até o tracejado do 1.9, e não a área pintada que é o que queremos. Sabendo disso, e que a área total abaixo da curva vale 1, para obter a área pintada basta subtrair 1 do valor de Z na tabela:

```
probabilidade = 0.9767 # Valor da área branca

probabilidade = 1 - 0.9767
probabilidade
```

```
0.023299999999999987
```

3.7.3.1.1. Ou seja, temos 2,32% de chance de ter alguém com mais de 1.9 de altura.

3.7.4. Solução 2: Scipy

3.7.4.1. Segue o mesmo raciocínio, 1 – norm.cdf(Z):

```
probabilidade = 1 - norm.cdf(Z)
probabilidade
```

```
0.02275013194817921
```

3.7.4.2. O scipy é inteligente e sabe que se a gente passar um número negativo pra ele estamos querendo calcular a área igual a desse caso:

```
probabilidade = norm.cdf(-Z)
probabilidade
```

```
0.02275013194817921
```

3.8. O que aprendemos:

3.8.1. Os conceitos básicos da distribuição normal de probabilidades;

3.8.2. A trabalhar com a tabela padronizada Z;

3.8.3. A obter as probabilidades utilizando a distribuição normal em um conjunto de situações.

4. Aula 4 – Técnicas de Amostragem:

4.1. **População**: Conjunto de todos os elementos de interesse em um estudo. Diversos elementos podem compor uma população, por exemplo: pessoas, idades, alturas, carros etc. Com relação ao tamanho, as populações podem ser limitadas (populações finitas) ou ilimitadas (populações infinitas).

4.1.1. Populações Finitas: Permitem a contagem de seus elementos. Como exemplos temos o número de funcionário de uma empresa, a quantidade de alunos em uma escola etc.

4.1.2. Populações Infinitas: Não é possível contar seus elementos. Como exemplos temos a quantidade de porções que se pode extrair da água do mar para uma análise, temperatura medida em cada ponto de um território etc.

4.1.2.1. **Quando os elementos de uma população puderem ser contados, porém apresentando uma quantidade muito grande, assume-se a população como infinita.**

4.2. **Amostra**: Subconjunto representativo da população.

4.2.1. Os atributos numéricos de uma população como sua média, variância e desvio padrão, são conhecidos como parâmetros. O principal foco da inferência estatística é justamente gerar estimativas e testar hipóteses sobre os parâmetros populacionais utilizando as informações de amostras.

4.3. **Quando utilizar uma amostra**:

4.3.1. Populações Infinitas: O estudo não chegaria nunca ao fim. Não é possível investigar todos os elementos da população.

4.3.2. Testes Destrutivos: Estudos onde os elementos avaliados são totalmente consumidos ou destruídos. Exemplo: testes de vida útil, testes de segurança contra colisões em automóveis.

4.3.3. Resultados Rápidos: Pesquisas que precisam de mais agilidade na divulgação. Exemplo: pesquisas de opinião, pesquisas que envolvam problemas de saúde pública.

4.3.4. Custos Elevados: Quando a população é finita, mas muito numerosa, o custo de um censo pode tornar o processo inviável.

4.4. **Amostragem Aleatória Simples:** É uma das principais maneiras de se extrair uma amostra de uma população. A exigência fundamental deste tipo de

abordagem é que cada elemento da população tenha as mesmas chances de ser selecionado para fazer parte da amostra.

4.4.1. Para pegar uma amostra utilizamos o df.sample(n = n_amostra, Random_state = n) do pandas. O segundo parâmetro é uma seed para sempre obtermos a mesma aleatorização, ou seja, sempre com as mesmas amostras:

```
amostra = dados.sample(n = 100, random_state = 101)
amostra
```

	UF	Sexo	Idade	Cor	Anos de Estudo	Renda	Altura
29042	29	0	39	8	5	480	1.719128
62672	43	0	55	2	6	250	1.639205
29973	29	1	36	2	12	788	1.654122
22428	26	0	46	8	8	1680	1.622450
55145	41	0	37	2	9	2500	1.625268
...	...	...	...	...	...	...	...
40245	32	0	35	2	12	2800	1.668898
30997	29	1	34	4	14	1924	1.795315
15094	23	0	60	4	1	130	1.682966
48788	35	1	39	2	7	460	1.777723
34891	31	0	42	2	3	1200	1.764392

100 rows × 7 columns

4.4.2. Ao fazermos a comparação da frequência do sexo dos indivíduos da nossa população e da nossa amostra, podemos observar que é praticamente a mesma:

```
dados.Sexo.value_counts(normalize = True)
```

```
0    0.692998
1    0.307002
Name: Sexo, dtype: float64
```

```
amostra.Sexo.value_counts(normalize = True)
```

```
0    0.69
1    0.31
Name: Sexo, dtype: float64
```

4.4.2.1. Podemos ainda alterar o tamanho da nossa amostra e, apesar de variar um pouco os resultados, continuam estando ao redor da população:

```
amostra = dados.sample(n = 1000, random_state = 101)
dados.Sexo.value_counts(normalize = True)
```

```
0    0.692998
1    0.307002
Name: Sexo, dtype: float64
```

```
amostra.Sexo.value_counts(normalize = True)
```

```
0    0.706
1    0.294
Name: Sexo, dtype: float64
```

4.5. **Amostragem Estratificada:** É uma melhoria do processo de amostragem aleatória simples. Neste método é proposta a divisão da população em subgrupos de elementos com características similares, ou seja, grupos mais homogêneos. Com estes subgrupos separados, aplica-se a técnica de amostragem aleatória simples dentro de cada subgrupo individualmente.

4.6. **Amostragem por Conglomerados:** Também visa melhorar o critério de amostragem aleatória simples. Na amostragem por conglomerados são também criados subgrupos, porém não serão homogêneas como na amostragem estratificada. Na amostragem por conglomerados os subgrupos serão heterogêneos, onde, em seguida, serão aplicadas a amostragem aleatória simples ou estratificada. Um exemplo bastante comum de aplicação deste tipo de técnica é na divisão da população em grupos territoriais, onde os elementos investigados terão características bastante variadas.

4.7. O que aprendemos:

4.7.1. Os conceitos de população e amostra;

4.7.2. A identificação de populações finita e infinita;

4.7.3. Quando utilizar a técnica de amostragem em um estudo;

4.7.4. Técnicas de seleção de amostra, como:

4.7.4.1. Amostragem aleatória simples;

4.7.4.2. Amostragem estratificada;

4.7.4.3. Amostragem por conglomerados.

5. Aula 5 – Nível e Intervalo de Confiança:

5.1. Estimação: Estimativa dos parâmetros de uma população a partir dos dados de uma amostra.

5.2. Problema para resolver: Suponha que os pesos dos sacos de arroz de uma indústria alimentícia se distribuem aproximadamente como uma normal de desvio padrão populacional igual a 150 g. Selecionada uma amostra aleatória de 20 sacos de um lote específico, obteve-se um peso médio de 5.050 g. Construa um intervalo de confiança para a média populacional assumindo um nível de significância de 5%.

5.3. É a forma de se fazer suposições generalizadas sobre os parâmetros de uma população tendo como base as informações de uma amostra.

5.3.1. Parâmetros são os atributos numéricos de uma população, tal como a média, desvio padrão etc.

5.3.2. Estimativa é o valor obtido para determinado parâmetro a partir dos dados de uma amostra da população.

5.4. **Teorema do Limite Central**: O Teorema do Limite Central afirma que, com o aumento do tamanho da amostra, a distribuição das médias amostrais se aproxima de uma distribuição normal com média igual à média da população e desvio padrão igual ao desvio padrão da variável original dividido pela raiz quadrada do tamanho da amostra. Este fato é assegurado para n maior ou igual a 30. Cálculo:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

5.4.1. O desvio padrão das médias amostrais é conhecido como erro padrão da média.

5.4.2. O Teorema do Limite Central afirma que, com o aumento do tamanho da amostra, a distribuição das médias amostrais se aproxima de uma distribuição normal com média igual à média da população e desvio padrão igual ao desvio padrão da variável original dividido pela raiz quadrada do tamanho da amostra. Este fato é assegurado para n maior ou igual a 30.

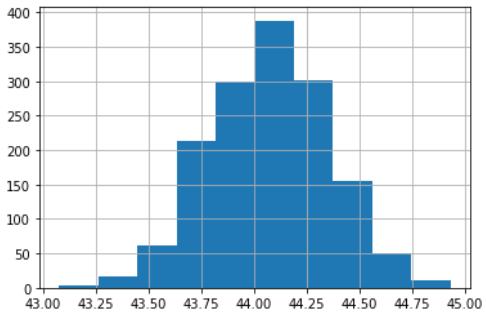
5.4.3. Para comprovar fizemos um df com 2k observações e 1.5k amostras da idade dos nossos dados e chamamos isso de amostras:

```
n = 2000
total_de_amostras = 1500
amostras = pd.DataFrame()
for i in range(total_de_amostras):
  _ = dados.Idade.sample(n)
```

```
  _.index = range(0, len(_))
  amostras['Amostra_' + str(i)] = _

amostras
```

	Amostra_0	Amostra_1	Amostra_2	Amostra_3	Amostra_4	Amostra_5	Amostra_6	Amostra_7	Amostra_8
0	59	67	50	58	44	60	64	23	27
1	65	44	35	29	45	37	29	37	43
2	31	31	39	59	49	52	41	53	47

```
amostras.mean().hist()
```



5.4.4. Se assemelhando bastante a uma distribuição normal.

5.4.5. O Teorema do Limite Central afirma que, com o aumento do tamanho da amostra, a distribuição das médias amostrais se aproxima de uma distribuição normal com média igual à média da população e desvio padrão igual ao desvio padrão da variável original dividido pela raiz quadrada do tamanho da amostra. Este fato é assegurado para n maior ou igual a 30.

5.4.6. Para comprovar tiramos a média das médias da nossa amostra e a média da nossa população:

```
amostras.mean()
```

```
Amostra_0       43.9725
Amostra_1       44.3185
Amostra_2       43.9770
Amostra_3       44.0165
Amostra_4       43.6130
                   ...
Amostra_1495    44.3350
Amostra_1496    43.9340
Amostra_1497    44.0650
Amostra_1498    44.2280
Amostra_1499    44.2055
Length: 1500, dtype: float64
```

```
amostras.mean().mean()
```

```
44.073457000000005
```

```
dados.Idade.mean()
```

```
44.07142113482561
```

5.4.7. O Teorema do Limite Central afirma que, com o aumento do tamanho da amostra, a distribuição das médias amostrais se

aproxima de uma distribuição normal com média igual à média da população e desvio padrão igual ao desvio padrão da variável original dividido pela raiz quadrada do tamanho da amostra. Este fato é assegurado para n maior ou igual a 30.

```
amostras.mean().std()
```

```
0.2814755453666024
```

```
dados.Idade.std() / np.sqrt(n)
```

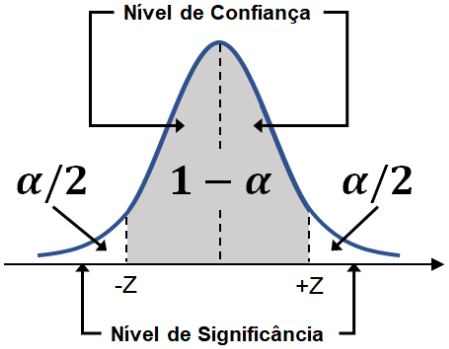
```
0.2790743302740527
```

5.5. Níveis de Confiança e significância:

5.5.1. O nível de confiança ($1-\alpha$) representa a probabilidade de acerto da estimativa. De forma complementar o nível de significância (α) expressa a probabilidade de erro da estimativa.

5.5.2. O nível de confiança representa o grau de confiabilidade do resultado da estimativa estar dentro de determinado intervalo. Quando fixamos em uma pesquisa um nível de confiança de 95%, por exemplo, estamos assumindo que existe uma probabilidade de 95% dos resultados da pesquisa representarem bem a realidade, ou seja, estarem corretos.

5.5.3. O nível de confiança de uma estimativa pode ser obtido a partir da área sob a curva normal como ilustrado na figura abaixo.



5.6. **Erro Inferencial**: O erro inferencial é definido pelo desvio padrão das médias amostrais $\sigma\bar{x}$ e pelo nível de confiança determinado para o processo. Cálculo:

$$e = z\frac{\sigma}{\sqrt{n}}$$

5.7. **Intervalo de Confiança**:

5.7.1. Intervalo de confiança para a média da população:

5.7.1.1. Com desvio padrão populacional conhecido:

$$\mu = \bar{x} \pm z\frac{\sigma}{\sqrt{n}}$$

5.7.1.2. Com desvio padrão populacional desconhecido:

$$\mu = \bar{x} \pm z\frac{s}{\sqrt{n}}$$

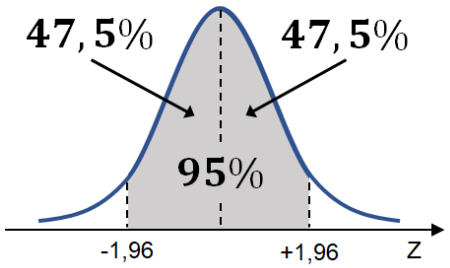
5.7.2. Exemplo: Suponha que os pesos dos sacos de arroz de uma indústria alimentícia se distribuem aproximadamente como uma normal de desvio padrão populacional igual a 150 g. Selecionada uma amostra aleatória de 20 sacos de um lote específico, obteve-se um peso médio de 5.050 g. Construa um intervalo de confiança para a média populacional assumindo um nível de significância de 5%.

5.7.2.1. media_amostral = 5050;

5.7.2.2. significancia = 0.05;

5.7.2.3. confianca = 1 – significância = 0.95;

5.7.2.4. Obtendo Z:




```
0.95/2
```

```
0.475
```

```
0.5 + (0.95/2)
```

```
0.975
```

```
1.9 + 0.06
```

```
1.96
```

```
z = norm.ppf(0.975)

z
```

```
1.959963984540054
```

5.7.3. Valores de z para os níveis de confiança mais utilizados:

Nível de confiança	Valor da área sob a curva normal	z
90%	0,95	1,645
95%	0,975	1,96
99%	0,995	2,575

5.7.4. Obtendo $\sigma_{\bar{x}}$:

```
desvio_padrao = 150
n = 20
raiz_de_n = np.sqrt(n)
raiz_de_n
```

```
4.47213595499958
```

```
sigma = desvio_padrao / raiz_de_n
sigma
```

```
33.54101966249684
```

5.7.5. Obtendo *e*:

```
e = z * sigma
e
```

```
65.73919054324361
```

5.7.6. Solução 1: Manual

```
intervalo = (
    media_amostral - e,
    media_amostral + e
)
intervalo
```

```
(4984.260809456757, 5115.739190543243)
```

5.7.7. Solução 2: Python

```
norm.interval(alpha = 0.95, loc = media_amostral, scale = sigma)
```

```
(4984.260809456757, 5115.739190543243)
```

5.8. O que aprendemos:

5.8.1. A conceituação de parâmetros e de estimativa;

5.8.2. O teorema do limite central;

5.8.3. Níveis de confiança e de significância;

5.8.4. A obter a margem de erro de um experimento;

5.8.5. A obter intervalos de confiança para uma estimativa pontual (estimação intervalar).

6. Aula 6 – Calculando o Tamanho da Amostra:

6.1. **População infinita**: Problema: Estamos estudando o rendimento mensal dos chefes de domicílios com renda até R$5.000,00 no Brasil. Nosso supervisor determinou que o erro máximo em relação à média seja de R$10,00. Sabemos que o desvio padrão populacional deste grupo de trabalhadores é de R$1.082,79. Para um nível de confiança de 95%, qual deve ser o tamanho da amostra de nosso estudo?

$$e = z\frac{\sigma}{\sqrt{n}}$$

6.1.1. Desvio padrão conhecido:

$$n = \left(z\frac{\sigma}{e}\right)^2$$

6.1.2. Desvio padrão desconhecido:

$$n = \left(z\frac{s}{e}\right)^2$$

6.1.3. Onde:

6.1.3.1. Z = variável normal padronizada;

6.1.3.2. σ = desvio padrão populacional;

6.1.3.3. s = desvio padrão amostral;

6.1.3.4. e = erro inferencial.

6.1.4. Observações:

6.1.4.1. O desvio padrão (σ ou s) e o erro (e) devem estar na mesma unidade de medida;

6.1.4.2. Quando o erro (e) for representado em termos percentuais, deve ser interpretado como um percentual relacionado à média.

6.1.5. Seguindo o problema:

6.1.5.1. z = norm.ppf(0.975) = 1.9599;

6.1.5.2. sigma = 3323.39;

6.1.5.3. e = 100;

6.1.5.4. n = (z * (sigma/e)) ** 2 = 4242.8609.

6.2. **População finita**: Problema: Em um lote de 10.000 latas de refrigerante foi realizada uma amostra aleatória simples de 100 latas e foi obtido o desvio padrão amostral do conteúdo das latas igual a 12 ml. O fabricante estipula um erro máximo sobre a média populacional de apenas 5 ml. Para garantir um nível de confiança de 95% qual o tamanho de amostra deve ser selecionado para este estudo?

6.2.1. Desvio padrão conhecido:

$$n = \frac{z^2 \sigma^2 N}{z^2 \sigma^2 + e^2 (N-1)}$$

6.2.2. Desvio padrão desconhecido:

$$n = \frac{z^2 s^2 N}{z^2 s^2 + e^2 (N-1)}$$

6.2.3. Onde:

6.2.3.1. N = tamanho da população;

6.2.3.2. z = variável normal padronizada;

6.2.3.3. σ = desvio padrão populacional;

6.2.3.4. s = desvio padrão amostral;

6.2.3.5. e = erro inferencial.

6.2.4. Problema:

6.2.4.1. N = 10000;

6.2.4.2. z = norm.ppf((0.5 + (0.95 / 2))) = 1.9599;

6.2.4.3. s = 12;

6.2.4.4. e = 5;

6.2.4.5. n = ((z ** 2) * (s ** 2) * (N)) / (((z ** 2) * (s ** 2)) + ((e ** 2) * (N - 1))) ≈ **22**.

6.3. O que aprendemos:

6.3.1. A determinação do tamanho de uma amostra, para garantir que esta seja representativa da população;

6.3.2. O cálculo de tamanho de amostra para variáveis quantitativas finitas e infinitas.

7. Aula 7 – Resumo e Projeto Final:

7.1. Exemplo de rendimento médio: Estamos estudando o rendimento mensal dos chefes de domicílios com renda até R$5.000,00 no Brasil. Nosso supervisor determinou que o erro máximo em relação à média seja de R$10,00. Sabemos que o desvio padrão populacional deste grupo de trabalhadores é de R$1.082,79 e que a média populacional é de R$1.426,54. Para um nível de confiança de 95%, qual deve ser o tamanho da amostra de nosso estudo? Qual o intervalo de confiança para a média considerando o tamanho de amostra obtido?

7.1.1. Construindo o dataset conforme especificado pelo problema:

7.1.1.1. renda_5000 = dados.query('Renda <= 5000').Renda;

7.1.1.2. sigma = renda_5000.std() = 1082.794;

7.1.1.3. media = renda_5000.mean() = 1426.5372.

7.1.2. Calculando o tamanho da amostra:

7.1.2.1. z = norm.ppf(.975);

7.1.2.2. e = 10;

7.1.2.3. n = (z * (sigma / e)) ** 2 ≈ 45039.

7.1.3. Calculando o intervalo de confiança para a média:

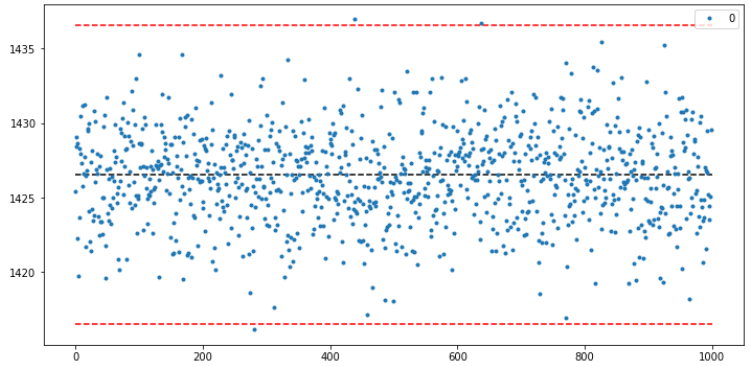
7.1.3.1. intervalo = norm.interval(alpha = 0.95, loc = media, scale = (sigma / np.sqrt(n))) =

7.1.3.2. (1416.5372144947232, 1436.5372144947232)

7.1.4. Realizando uma prova gráfica:

```
import matplotlib.pyplot as plt


tamanho_simulacao = 1000

medias = [renda_5000.sample(n = n).mean() for i in range(1, tamanho
_simulacao)]
medias = pd.DataFrame(medias)
```

```
ax = medias.plot(style = '.')
ax.figure.set_size_inches(12, 6)
ax.hlines(y = media, xmin = 0, xmax = tamanho_simulacao, colors='bl
ack', linestyles='dashed')
ax.hlines(y = intervalo[0], xmin = 0, xmax = tamanho_simulacao, col
ors='red', linestyles='dashed')
ax.hlines(y = intervalo[1], xmin = 0, xmax = tamanho_simulacao, col
ors='red', linestyles='dashed')
ax
```



7.2. O que aprendemos:

7.2.1. A entender os intervalos de confiança, com a realização de simulações de processos de amostragem com prova gráfica;

7.2.2. A aplicação das técnicas desenvolvidas no treinamento para a solução de problemas reais;

7.2.3. O dimensionamento de um processo de pesquisa de campo, com controle de custos e margem de erro.

Parte III – Testes de Hipóteses:

Link para o drive:
https://drive.google.com/drive/folders/1WZH8jLiIloHR0dPd0glkRMUhvMhOwLx5

1. **Aula 1 – Teste De Normalidade e As Etapas De Um Teste:**

1.1. Testes de hipóteses: Testes estatísticos são regras de decisão que permitem avaliar a razoabilidade das hipóteses feitas sobre os parâmetros populacionais e aceitá-las ou rejeitá-las como provavelmente verdadeiras ou falsas tendo como base uma amostra.

1.1.1. Teste de normalidade: Importar a biblioteca normaltest from scipy.stats. A função normaltest testa a hipótese nula H0 de que a amostra é proveniente de uma distribuição normal.

1.1.2. Nível de confiança é a probabilidade de dar certo e a de significância é a de dar errado, ou seja, 0.05 de significância (chance de dar errado) é equivalente a 0.95 de confiança (chance de dar certo).

1.1.3. Rejeitamos H0, ou seja, a hipótese de que de que a amostra é proveniente de uma distribuição normal quando o nosso valor de p é menor que 0,05.

1.1.4. Fazendo o teste. O normaltest() nos devolve uma tupla, então separamos em 2 variáveis:

```
stat_test, p_valor = normaltest(dados.Renda)
print(stat_test, p_valor)
```

```
152380.75803960307 0.0
```

```
p_valor <= significancia
```

```
True
```

1.1.4.1. Ou seja, as rendas dos nossos dados não são normais.

1.1.5. Fazendo o mesmo com altura:

```
stat_test, p_valor = normaltest(dados.Altura)
print(stat_test, p_valor)
```

```
0.19973093957002253 0.9049591541967501
```

```
p_valor <= significancia
```

```
False
```

1.1.5.1. Ou seja, aceitamos H0, portanto, nossa distribuição é normal para altura.

1.2. Etapas Básicas de um teste:

1.2.1. **Passo 1: Formulação das hipóteses H0 e H1**:

1.2.1.1. De maneira geral, o alvo do estudo deve ser formulado como a hipótese alternativa H1.

1.2.1.2. A hipótese nula sempre afirma uma igualdade ou propriedade populacional, e H1 a desigualdade que nega H0.

1.2.1.3. No caso da hipótese nula H0 a igualdade pode ser representada por uma igualdade simples "=" ou por "≥" e "≤". Sempre complementar ao estabelecido pela hipótese alternativa.

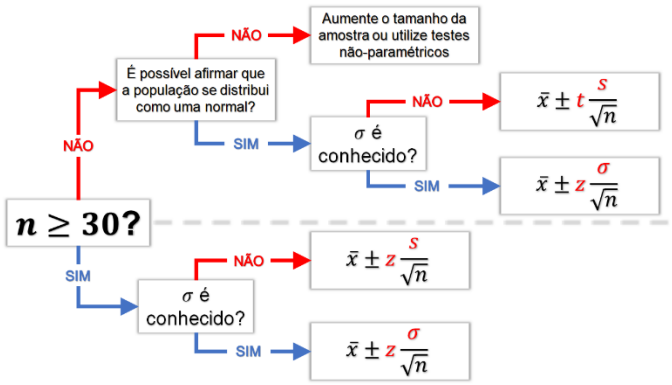
1.2.1.4. A hipótese alternativa H1 deve definir uma desigualdade que pode ser uma diferença simples "≠" ou dos tipos ">" e "<".

1.2.2. **Passo 2: Escolha da distribuição amostral adequada**:

1.2.2.1. Quando o tamanho da amostra tiver 30 elementos ou mais, deve-se utilizar a distribuição normal, como estabelecido pelo teorema do limite central.

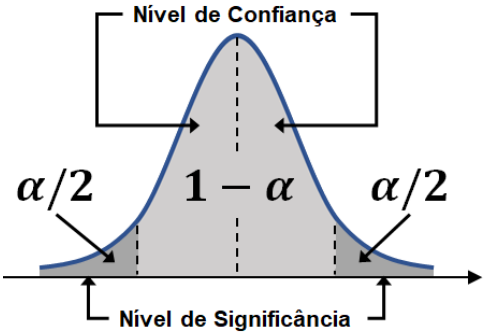
1.2.2.2. Para um tamanho de amostra menor que 30 elementos, e se pudermos afirmar que a população se distribui aproximadamente como uma normal e o desvio padrão populacional for conhecido, deve-se utilizar a distribuição normal.

1.2.2.3. Para um tamanho de amostra menor que 30 elementos, e se pudermos afirmar que a população se distribui aproximadamente como uma normal e o desvio padrão populacional for desconhecido, deve-se utilizar a distribuição t de Student.

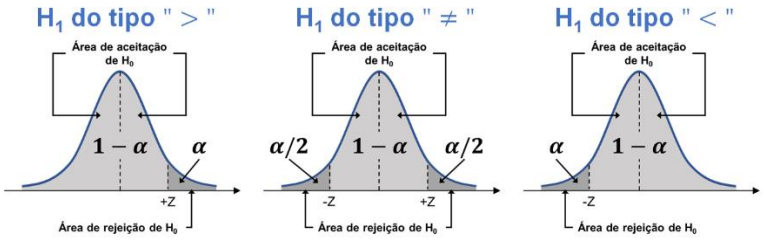


1.2.3. **Passo 3: Fixação da Significância do teste (alpha), que define as regiões de aceitação e rejeição das hipóteses (os valores mais frequentes são 10%, 5% e 1%:**

1.2.3.1. O nível de confiança (1−α) representa a probabilidade de acerto da estimativa. De forma complementar o nível de significância (α) expressa a probabilidade de erro da estimativa.



1.2.3.2. O nível de confiança representa o grau de confiabilidade do resultado da estimativa estar dentro de determinado intervalo. Quando fixamos em uma pesquisa um nível de confiança de 95%, por exemplo, estamos assumindo que existe uma probabilidade de 95% dos resultados da pesquisa representarem bem a realidade, ou seja, estarem corretos.



1.2.4. **Passo 4: Cálculo da Estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:**

1.2.4.1. Nos testes paramétricos, distância relativa entre a estatística amostral e o valor alegado como provável.

1.2.4.2. Neste passo são obtidas as estatísticas amostrais necessárias à execução do teste (média, desvio-padrão, graus de liberdade etc.)

1.2.5. **Passo 5: Aceitação ou Rejeição da Hipótese Nula**:

1.2.5.1. No caso de o intervalo de aceitação conter a estatística-teste, aceita-se H0 como estatisticamente válido e rejeita-se H1 como tal.

1.2.5.2. No caso de o intervalo de aceitação não conter a estatística-teste, rejeita-se H0 e aceita-se H1 como provavelmente verdadeira.

1.2.5.3. A aceitação também se verifica com a probabilidade de cauda (p-valor): se maior que α, aceita-se H0.

1.3. O que aprendemos:

1.3.1. A preparar um ambiente de trabalho no Colaboratory;

1.3.2. A executar um teste de normalidade com as ferramentas da biblioteca Scipy;

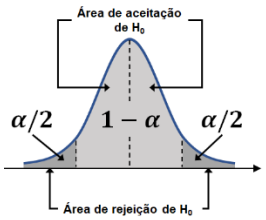
1.3.3. As cinco principais etapas para elaboração de um teste de hipóteses.

2. Aula 2 – Teste Bicaudal e Entendendo o Valor de p:

2.1. Testes Paramétricos: Quando um teste assume determinadas premissas sobre como os parâmetros de uma população se distribuem, estamos trabalhando com Testes Paramétricos.

2.2. Problema: A empresa Suco Bom produz sucos de frutas em embalagens de 500 ml. Seu processo de produção é quase todo automatizado e as embalagens de sucos são preenchidas por uma máquina que às vezes apresenta um certo desajuste, levando a erros no preenchimento das embalagens para mais ou menos conteúdo. Quando o volume médio cai abaixo de 500 ml, a empresa se preocupa em perder vendas e ter problemas com os órgãos fiscalizadores. Quando o volume passa de 500 ml, a empresa começa a se preocupar com prejuízos no processo de produção. O setor de controle de qualidade da empresa Suco Bom extrai, periodicamente, amostras de 50 embalagens para monitorar o processo de produção. Para cada amostra, é realizado um teste de hipóteses para avaliar se o maquinário se desajustou. A equipe de controle de qualidade assume um nível de significância de 5%. Suponha agora que uma amostra de 50 embalagens foi selecionada e que a média amostral observada foi de 503,24 ml. Esse valor de média amostral é suficientemente maior que 500 ml para nos fazer rejeitar a hipótese de que a média do processo é de 500 ml ao nível de significância de 5%?

2.3. Teste Bicaudal: O teste bicaudal é muito utilizado em testes de qualidade, como o apresentado em nosso problema acima. Outro exemplo é a avaliação de peças que devem ter um encaixe perfeito (porcas e parafusos, chaves e fechaduras).



2.4. Solucionando o problema:

```
amostra = [509, 505, 495, 510, 496, 509, 497, 502, 503, 505,
           501, 505, 510, 505, 504, 497, 506, 506, 508, 505,
           497, 504, 500, 498, 506, 496, 508, 497, 503, 501,
           503, 506, 499, 498, 509, 507, 503, 499, 509, 495,
           502, 505, 504, 509, 508, 501, 505, 497, 508, 507]

amostra = pd.DataFrame(amostra, columns=['Amostra'])
```

2.4.1. media_amostra = amostra.mean()[0] = 503.24;

2.4.2. desvio_padrao_amostra = amostra.std()[0] = 4.4838;

2.4.3. media = 500;

2.4.4. significancia = 0.05;

2.4.5. confianca = 1 – significância;

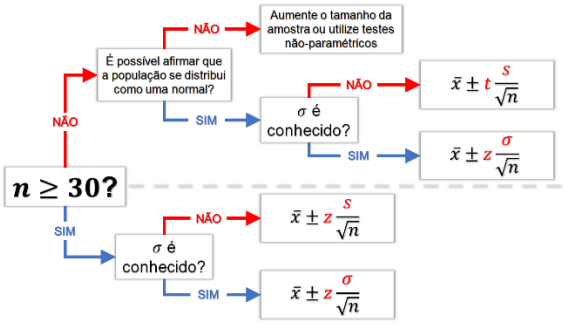
2.4.6. n = 50.

2.5. Passo 1: Formulação H0 e H1:

2.5.1. **Lembre-se, a hipótese nula sempre contém a alegação de igualdade.**

2.5.2. H0 = 500 e H1 ≠ 500;

2.6. Passo 2: escolha da distribuição amostral adequada:



2.6.1. Tamanho maior que 30? R: Sim;

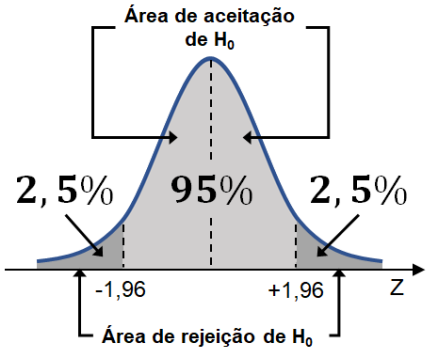
2.6.2. Desvio padrão populacional conhecido (sigma)? R: Não;

2.7. Passo 3: Fixação da Significância do teste (alpha):

```
from scipy.stats import norm
```

2.7.1. probabilidade = (0.5 + (confianca / 2)) = 0.975;

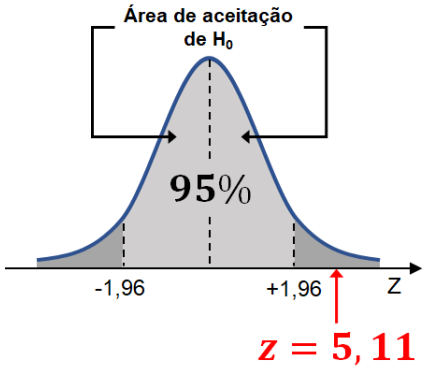
2.7.2. z_alpha_2 = norm.ppf(probabilidade) = 1.95996;



2.8. Passo 4: Cálculo da estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:

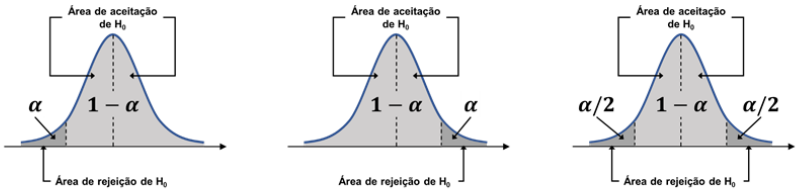
$$z = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$

2.8.1. z = (media_amostra - media) / (desvio_padrao_amostra / np.sqrt(n)) = 5.1095.



2.8.2. Ou seja, rejeitamos que a média é igual a 500 a um nível de significância de 5%.

2.9. Passo 5: Aceitação ou rejeição da hipótese:



	Teste da Cauda Inferior	Teste da Cauda Superior	Teste Bicaudal
Hipóteses	$H_0: \mu \geq \mu_0$ $H_1: \mu < \mu_0$	$H_0: \mu \leq \mu_0$ $H_1: \mu > \mu_0$	$H_0: \mu = \mu_0$ $H_1: \mu \neq \mu_0$
Estatística de Teste	$z = \frac{\bar{x} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$ ou $t = \frac{\bar{x} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$		
Regras de Rejeição de H0			
Valor Crítico z	Rejeitar se $z \leq -z_\alpha$	Rejeitar se $z \geq z_\alpha$	Rejeitar se $z \leq -z_{\alpha/2}$ ou se $z \geq z_{\alpha/2}$
Valor Crítico t	Rejeitar se $t \leq -t_\alpha$	Rejeitar se $t \geq t_\alpha$	Rejeitar se $t \leq -t_{\alpha/2}$ ou se $t \geq t_{\alpha/2}$
Valor p	Rejeitar se $p\_value \leq \alpha$		

2.9.1. z <= -z_alpha_2 = False, mas:

2.9.2. z >= -z_alpha_2 = True, ou seja, rejeita H0.

2.10. Critério do valor de p:

2.10.1. Bicaudal: Rejeita H0 se p<= alpha.

```
p_valor = 2 * (1 - norm.cdf(z))
p_valor
```

```
3.2291031715203644e-07
```

```
p_valor  = 2 * (norm.sf(z))
p_valor
```

```
3.229103172445718e-07
```

2.10.2. Podemos fazer toda essa conta de z bem mais fácil de 2 jeitos:

```
from statsmodels.stats.weightstats import ztest
ztest(x1=amostra, value=media)
```

```
(array([5.10955978]), array([3.22910317e-07]))
```

```
from statsmodels.stats.weightstats import DescrStatsW
test = DescrStatsW(amostra)
test.ztest_mean(value=media)
```

```
(array([5.10955978]), array([3.22910317e-07]))
```

2.11. O que aprendemos:

2.11.1. A aplicação de um teste paramétrico bicaudal;

2.11.2. A execução dos passos de um teste de hipóteses;

2.11.3. As regras de rejeição da hipótese nula de um teste;

2.11.4. A entender e calcular o p-valor de um teste;

2.11.5. A aplicação de um teste z com as ferramentas do Python.

3. Aula 3 - Distribuição t De Student e o Teste Unicaudal:

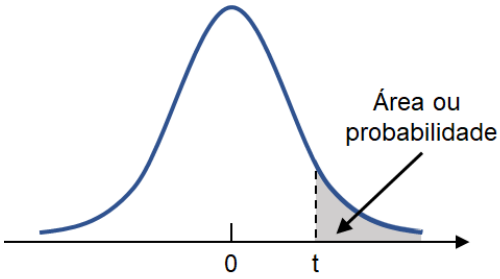
3.1. Problema: Um famoso fabricante de refrigerantes alega que uma lata de 350 ml de seu principal produto contém, no máximo, 37 gramas de açúcar. Esta alegação nos leva a entender que a quantidade média de açúcar em uma lata de refrigerante deve ser igual ou menor que 37g. Um consumidor desconfiado e com conhecimentos em inferência estatística resolve testar a alegação do fabricante e seleciona, aleatoriamente, em um conjunto de estabelecimentos distintos, uma amostra de 25 latas do refrigerante em questão. Utilizando o equipamento correto o consumidor obteve as quantidades de açúcar em todas as 25 latas de sua amostra. Assumindo que essa população se distribua aproximadamente como uma normal e considerando um nível de significância de 5%, é possível aceitar como válida a alegação do fabricante?

3.2. Gerando a tabela t-student:

```
import pandas as pd

from scipy.stats import t as t_student

tabela_t_student = pd.DataFrame(
    [],
    index = [i for i in range(1, 31)],
    columns = [1 / 100 for i in range (10, 0, -1)]
)

for index in tabela_t_student.index:
  for column in tabela_t_student.columns:
    tabela_t_student.loc[index, column] = t_student.ppf(1- float (c
olumn) / 2, index)

index= [('Graus de Liberdade (n - 1)', i) for i in range(1, 31)]
tabela_t_student.index = pd.MultiIndex.from_tuples(index)
```

```
columns = [("{0:0.3f}" .format(i / 100), "{0:0.3f}" .format((i / 10
0) / 2)) for i in range (10, 0, -1)]
tabela_t_student.columns = pd.MultiIndex.from_tuples(columns)

tabela_t_student.rename_axis (['Bicaudal', 'Unicaudal' ], axis=1, i
nplace = True)

tabela_t_student
```

3.3. Área de probabilidade unicaudal:



3.4. Teste:

3.4.1. media = 37

3.4.2. significancia = 0.05

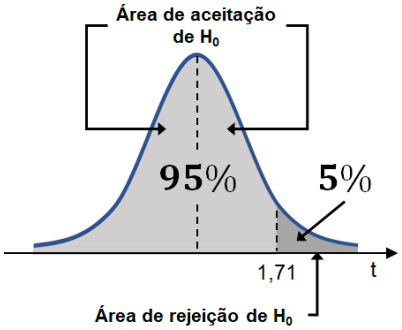
3.4.3. confianca = 1 - significancia

3.4.4. n = 25

3.4.5. grau_de_liberdade = n – 1

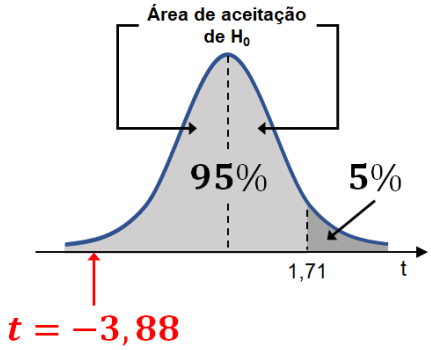
3.4.6. Obtendo t:

3.4.6.1. t_alpha = t_student.ppf(confianca, grau_de_liberdade) = 1,71



3.4.6.2. t = (media_amostra - media) / (desvio_padrao_amostra / np.sqrt(n)) = -3.87689

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$



3.4.7. Com python:

```
t, p_valor, df = test.ttest_mean(value=media, alternative='larger')
print(t[0])
print(p_valor[0])
print(df)
```

```
-3.8768931199520447
0.9996406170303819
24.0
```

3.5. O que aprendemos:

3.5.1. A distribuição t de Student;

3.5.2. A construir e consultar uma tabela t de Student;

3.5.3. A aplicação de um teste paramétrico unicaudal;

3.5.4. A definição de hipóteses e obtenção de áreas críticas para um teste unicaudal;

3.5.5. A aplicação de um teste t com as ferramentas do Python.

4. Aula 4 – Teste Para Duas Amostras:

4.1. Problema: Em nosso dataset temos os rendimentos dos chefes de domicílio obtidos da Pesquisa Nacional por Amostra de Domicílios - PNAD no ano de 2015. Um problema bastante conhecido em nosso país diz respeito a desigualdade de renda, principalmente entre homens e mulheres. Duas amostras aleatórias, uma de 500 homens e outra com 500 mulheres, foram selecionadas em nosso dataset. Com o objetivo de comprovar tal desigualdade,

teste a igualdade das médias entre estas duas amostras com um nível de significância de 1%.

4.2. Obtendo dados para resolver o problema:

4.2.1. homens = dados.query('Sexo == 0').sample(n = 500, random_state = 101).Renda

4.2.2. mulheres = dados.query('Sexo == 1').sample(n = 500, random_state = 101).Renda

4.2.3. media_amostra_M = mulheres.mean() = 1357.528;

4.2.4. desvio_padrao_amostra_M = mulheres.std() = 1569.9011;

4.2.5. media_amostra_H = homens.mean() = 2142.608;

4.2.6. desvio_padrao_amostra_H = homens.std() = 2548.05080;

4.2.7. significancia = 0.01;

4.2.8. confianca = 1 – significância = 0.99;

4.2.9. n_M = 500;

4.2.10. n_H = 500;

4.2.11. D_0 = 0.

4.3. Hipótese:

$\mu_1 \Rightarrow$ Média das rendas dos chefes de domicílios do sexo masculino

$\mu_2 \Rightarrow$ Média das rendas dos chefes de domicílios do sexo feminino

$$\begin{cases} H_0 : \mu_1 \leq \mu_2 \\ H_1 : \mu_1 > \mu_2 \end{cases}$$

ou

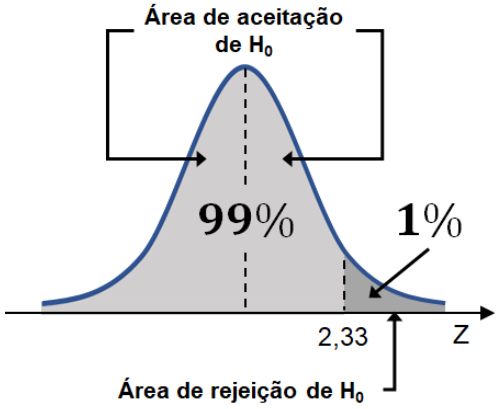
$$\begin{cases} H_0 : \mu_1 - \mu_2 \leq 0 \\ H_1 : \mu_1 - \mu_2 > 0 \end{cases}$$

Em testes que envolvam duas amostras com o emprego da tabela t de Student, o número de graus de liberdade será sempre igual a n1+n2−2.

4.4. Fixação da significância:

4.4.1. probabilidade = confiança = 0.99;

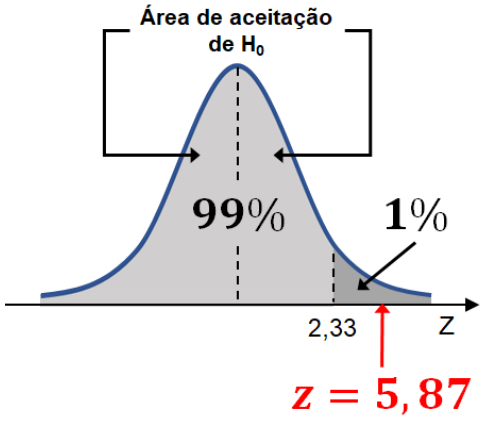
4.4.2. z_alpha = norm.ppf(probabilidade) = 2.33;



4.5. cálculo da estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:

$$z = \frac{(\bar{x_1} - \bar{x_2}) - D_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

4.5.1. z = ((media_amostra_M - media_amostra_H) - D_0) / (np.sqrt(((desvio_padrao_amostra_M ** 2) / n_M) + ((desvio_padrao_amostra_H ** 2) / n_H))) = -5.8656.



4.6. Rejeita ou aceita:

	Teste da Cauda Inferior	Teste da Cauda Superior	Teste Bicaudal
Hipóteses	$H_0: \mu_1 - \mu_2 \geq D_0$ $H_1: \mu_1 - \mu_2 < D_0$	$H_0: \mu_1 - \mu_2 \leq D_0$ $H_1: \mu_1 - \mu_2 > D_0$	$H_0: \mu_1 - \mu_2 = D_0$ $H_1: \mu_1 - \mu_2 \neq D_0$
Estatística de Teste	$z = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$ ou $t = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$		
Regras de Rejeição de H_0			
Valor Crítico z	Rejeitar se $z \leq -z_\alpha$	Rejeitar se $z \geq z_\alpha$	Rejeitar se $z \leq -z_{\alpha/2}$ ou se $z \geq z_{\alpha/2}$
Valor Crítico t	Rejeitar se $t \leq -t_\alpha$	Rejeitar se $t \geq t_\alpha$	Rejeitar se $t \leq -t_{\alpha/2}$ ou se $t \geq t_{\alpha/2}$
Valor p	Rejeitar se $p\_value \leq \alpha$		

4.7. Bibliotecas para estatísticas:

```
from statsmodels.stats.weightstats import DescrStatsW, CompareMeans
```

4.7.1. Utilizamos essas bibliotecas para fazer a análise de z e p com python:

```
test_H = DescrStatsW(homens)
test_M = DescrStatsW(mulheres)
test = test_H.get_compare(test_M)
z, p = test.ztest_ind(alternative='larger', value=0)
print(z, p)
```

```
5.865620057764754 2.2372867859458255e-09
```

4.7.2. Ou seja, recebemos o mesmo resultado/conclusão que antes:

```
p <= significancia
```

```
True
```

4.7.2.1. Com um nível de confiança de 99% rejeitamos H0, isto é, concluímos que a média das rendas dos chefes de domicílios do sexo masculino é maior que a média das rendas das chefes de domicílios do sexo feminino. Confirmando a alegação de desigualdade de renda entre os sexos.

4.7.3. Podemos ainda fazer com a biblioteca CompareMeans:

```
test = CompareMeans(test_H, test_M)
z, p = test.ztest_ind(alternative='larger', value=0)
print(z, p)
```

```
5.865620057764754 2.2372867859458255e-09
```

4.8. O que aprendemos:

4.8.1. A aplicação de teste de comparação entre médias de amostras diferentes;

4.8.2. A definição das hipóteses para testes entre duas amostras;

4.8.3. A aplicação de um teste z, para duas amostras, com as ferramentas do Python.

5. Aula 5 – Distribuição Qui-Quadrado:

5.1. Testes não paramétricos: O trabalho com pequenas amostras pode levar a não aceitação da validade do teorema central do limite e também na impossibilidade de fazer suposições sobre a distribuição da variável avaliada. Quando isso ocorre torna-se necessária a aplicação de testes não paramétricos. Nos testes não paramétricos, não fazemos hipóteses sobre a distribuição (de probabilidade) das quais as observações são extraídas.

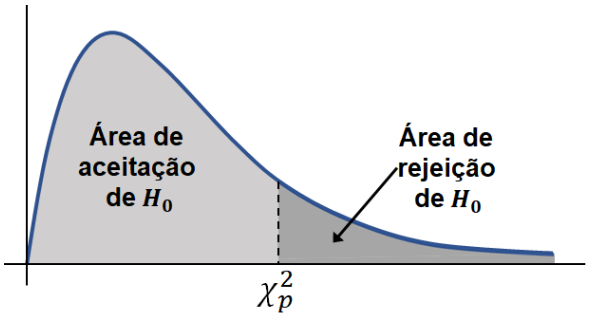
5.2. Problema: Antes de cada partida do campeonato nacional de futebol, as moedas utilizadas pelos árbitros devem ser verificadas para se ter certeza de que não são viciadas, ou seja, que não tendam para determinado resultado. Para isso um teste simples deve ser realizado antes de cada partida. Este teste consiste em lançar a moeda do jogo 50 vezes e contar as frequências de CARAS e COROAS obtidas. A tabela abaixo mostra o resultado obtido no experimento:

	CARA	COROA
Observado	17	33
Esperado	25	25

5.2.1. A um nível de significância de 5%, é possível afirmar que a moeda não é honesta, isto é, que a moeda apresenta uma probabilidade maior de cair com a face CARA voltada para cima?

5.3. Teste Qui-Quadrado (x^2): Também conhecido como teste de adequação ao ajustamento, seu nome se deve ao fato de utilizar uma variável estatística padronizada, representada pela letra grega qui (χ) elevada ao quadrado. A tabela com os valores padronizados e como obtê-la podem ser vistos logo abaixo. O teste do $\chi 2$ testa a hipótese nula de não haver diferença entre as frequências observadas de um determinado evento e as frequências que são

realmente esperadas para este evento. Os passos de aplicação do teste são bem parecidos aos vistos para os testes paramétricos.



5.4. Tabela com os valores de $\chi 2p$ em função dos graus de liberdade $(n-1)$ e de $p = P(\chi^2 \leq \chi^2_p)$.

5.5. Código para tabela x^2:

```
import pandas as pd
from scipy.stats import chi

tabela_t_chi_2 = pd.DataFrame(
    [],
    index = [i for i in range(1, 31)],
    columns = [0.005, 0.01, 0.025, 0.5, 0.75, 0.9, 0.975, 0.95, 0.9
9, 0.995]
)

for index in tabela_t_chi_2.index:
  for column in tabela_t_chi_2.columns:
    tabela_t_chi_2.loc[index, column] = '{0:0.4f}'.format(chi.ppf(f
loat(column), index) ** 2)

tabela_t_chi_2.index.name = 'Graus de Liberdade'
tabela_t_chi_2.rename_axis(['p'], axis = 1, inplace = True)

tabela_t_chi_2
```

5.6. Resolvendo o Problema:

5.6.1. Dados:

5.6.1.1. f_observada = [17, 33];

5.6.1.2. f_esperada = [25, 25];

5.6.1.3. significancia = 0.05;

5.6.1.4. confianca = 1 – significância;

5.6.1.5. k = 2 # Número de eventos possíveis;

5.6.1.6. graus_de_liberdade = k – 1.

5.6.2. Passo 1: H_0 e H_1:

5.6.2.1. H_0: Fcara = Fcoroa;

5.6.2.2. H_1: Fcara ≠ Fcoroa.

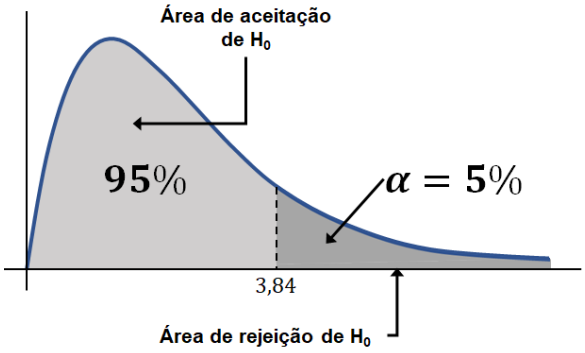
5.6.3. Passo 2: Fixação da Significância do Teste (α):

5.6.3.1. Não é necessário pois já fizemos nos dados do problema. Ao invés disso chamamos a tabela novamente e vamos obter o χ^2_α:

```
from scipy.stats import chi

tabela_t_chi_2[:3]
```

p	0.005	0.010	0.025	0.500	0.750	0.900	0.975	0.950	0.990	0.995
Graus de Liberdade										
1	0.0000	0.0002	0.0010	0.4549	1.3233	2.7055	5.0239	3.8415	6.6349	7.8794
2	0.0100	0.0201	0.0506	1.3863	2.7726	4.6052	7.3778	5.9915	9.2103	10.5966
3	0.0717	0.1148	0.2158	2.3660	4.1083	6.2514	9.3484	7.8147	11.3449	12.8382

5.6.3.2. chi_2_alpha = chi.ppf(confianca, graus_de_liberdade) ** 2 = 3.84145882



5.6.4. Passo 3: cálculo da estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:

$$\chi^2 = \sum_{i=1}^{k} \frac{(F_i^{Obs} - F_i^{Esp})^2}{F_i^{Esp}}$$

5.6.4.1. Onde:

5.6.4.2. F_i^{Obs} = frequência observada para o evento i;

5.6.4.3. F_i^{Esp} = frequência esperada para o evento i;

5.6.4.4. k = total de eventos possíveis.

5.6.4.5. É a somatória dos valores observados e esperados, ou seja, essa conta se repetirá para todos os valores observados e esperados:

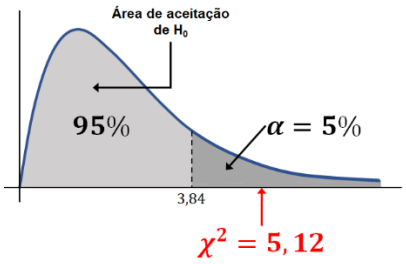
```
chi_2 = ((f_observada[0] - f_esperada[0]) ** 2 / f_esperada[0]) + (
(f_observada[1] - f_esperada[1]) ** 2 / f_esperada[1])
```

```
5.12
```

5.6.4.6. Não tem tanto problema repetir isso quando se tem poucos dados, mas quando esse número aumenta podemos simplificar fazendo um for:

```
chi_2 = 0

for i in range(k):
  chi_2 += (f_observada[i] - f_esperada[i]) ** 2 / f_esperada[i]

chi_2
```

```
5.12
```



5.6.5. Passo 4: Aceitação ou não de H_0:

Rejeitar H_0 se $\chi^2_{teste} > \chi^2_{\alpha}$

	Teste Qui-Quadrado
Hipóteses	$H_0: F_1 = F_2$ $H_1: F_1 \neq F_2$
Estatística de Teste	$\chi^2 = \sum_{i=1}^{k} \frac{(F_i^{Obs} - F_i^{Esp})^2}{F_i^{Esp}}$
Regras de Rejeição de H_0	
Valor Crítico χ	Rejeitar se $\chi^2 > \chi^2_{\alpha}$
Valor p	Rejeitar se $p\_value \leq \alpha$

```
chi_2 > chi_2_alpha
```

```
True
```

5.6.6. Conclusão: Com um nível de confiança de 95% rejeitamos a hipótese nula (H0) e concluímos que as frequências observadas e esperadas são discrepantes, ou seja, a moeda não é honesta e precisa ser substituída.

5.7. Aplicando o p-valor:

5.7.1. Precisamos fazer a raiz do chi^2 para conseguirmos o p, lembrando que rejeitamos H0 se p < significância, ou seja, 0,05:

```
raiz_chi_2 = np.sqrt(chi_2)
p_valor = chi.sf(raiz_chi_2, df=1)
p_valor
```

```
0.023651616655355978
```

5.7.2. Podemos facilitar TODO esse processo com 1 linha de código usando o chisquare do scipy.stats:

```
from scipy.stats import chisquare
chi_2, p_valor = chisquare(f_obs = f_observada, f_exp = f_esperada)
print(chi_2, p_valor)
```

```
5.12 0.023651616655356
```

5.8. O que aprendemos:

5.8.1. A distribuição Qui-quadrado;

5.8.2. A construir e consultar uma tabela de Qui-quadrado;

5.8.3. A aplicação do teste não paramétrico de Qui-quadrado;

5.8.4. O cálculo do p-valor com a distribuição Qui-quadrado.

6. Aula 6 – Teste de Wilcoxon e Mann-Whitney:

6.1. Teste Wilcoxon: Comparação de duas populações - amostras dependentes:

6.1.1. Empregado quando se deseja comparar duas amostras relacionadas, amostras emparelhadas. Pode ser aplicado quando se deseja testar a diferença de duas condições, isto é, quando um mesmo elemento é submetido a duas medidas.

6.2. Problema: Um novo tratamento para acabar com o hábito de fumar está sendo empregado em um grupo de 35 pacientes voluntários. De cada paciente testado foram obtidas as informações de quantidades de cigarros consumidos por dia antes e depois do término do tratamento. Assumindo um nível de confiança de 95% é possível concluir que, depois da aplicação do novo tratamento, houve uma mudança no hábito de fumar do grupo de pacientes testado?

6.3. Solução:

6.3.1. Dados:

```
fumo = {
    'Antes': [39, 25, 24, 50, 13, 52, 21, 29, 10, 22, 50, 15, 36, 3
9, 52, 48, 24, 15, 40, 41, 17, 12, 21, 47, 14, 55, 46, 22, 28, 23,
37, 17, 31, 49, 51],
    'Depois': [16, 8, 12, 0, 14, 16, 13, 12, 19, 17, 17, 2, 15, 10,
 20, 13, 0, 4, 16, 18, 16, 16, 9, 9, 18, 4, 17, 0, 11, 14, 0, 19, 2
, 9, 6]
}
```

6.3.1.1. significancia = 0.05;

6.3.1.2. confianca = 1 – significância = 0.95;

6.3.1.3. n = 25;

6.3.1.4. fumo = pd.DataFrame(fumo);

6.3.1.5. media_antes = fumo.Antes.mean() = 31.85714;

6.3.1.6. media_depois = fumo.Depois.mean() = 11.2;

6.3.2. Passo 1: H0 e H1:

$$H_0 : \mu_{antes} = \mu_{depois}$$
$$H_1 : \mu_{antes} > \mu_{depois}$$

6.3.3. Passo 2: escolha da distribuição amostral adequada:

6.3.3.1. O tamanho da amostra é maior que 20? Resp.: Sim;

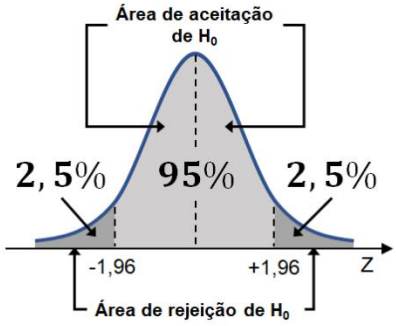
6.3.3.2. Usamos a tabela Z, se fosse não, utilizaríamos a de t-student.

6.3.4. Passo 3: fixação da significância do teste (α):

6.3.4.1. Obtendo $z_{\alpha/2}$:

6.3.4.1.1. probabilidade = (0.5 + (confianca / 2)) = 0.975;

6.3.4.1.2. z_alpha_2 = norm.ppf(probabilidade) = 1.96.



6.3.5. Passo 4 - cálculo da estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:

$$Z = \frac{T - \mu_T}{\sigma_T}$$

Onde

T = menor das somas de postos de mesmo sinal

$$\mu_T = \frac{n(n+1)}{4}$$

$$\sigma_T = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

6.3.6. Construindo a Tabela com os postos:

```
fumo['Dif'] = fumo.Depois - fumo.Antes
fumo['|Dif|'] = fumo.Dif.abs()
fumo.sort_values(by='|Dif|', inplace=True)
fumo['Posto'] = range(1, len(fumo) + 1)
posto = fumo[['|Dif|', 'Posto']].groupby(['|Dif|']).mean()
posto.reset_index(inplace=True)
fumo.drop(['Posto'], axis=1, inplace= True)
fumo = fumo.merge(posto, left_on='|Dif|', right_on='|Dif|', how='le
ft')
fumo['Posto (+)'] = fumo.apply(lambda x: x.Posto if x.Dif > 0 else
0, axis=1)
fumo['Posto (-
)'] = fumo.apply(lambda x: x.Posto if x.Dif < 0 else 0, axis=1)
fumo.drop(['Posto'], axis=1, inplace=True)
```

	Antes	Depois	Dif	|Dif|	Posto (+)	Posto (-)
0	13	14	1	1	1.5	0.0
1	17	16	-1	1	0.0	1.5
2	17	19	2	2	3.0	0.0
3	12	16	4	4	4.5	0.0
4	14	18	4	4	4.5	0.0
5	22	17	-5	5	0.0	6.0

6.3.7. Obter T: Menor das somas de postos de mesmo sinal:

6.3.7.1. T = min(fumo['Posto (+)'].sum(), fumo['Posto (-)'].sum()) = 22,0.

6.3.8. Obter uT:

$$\mu_T = \frac{n(n+1)}{4}$$

6.3.8.1. mu_t = (n * (n + 1)) / 4 = 315,0.

6.3.9. Obter sigmaT:

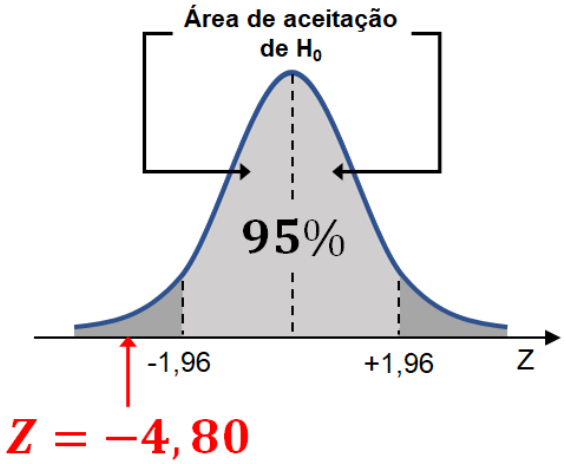
$$\sigma_T = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

6.3.9.1. sigma_t = np.sqrt((n * (n + 1)) * ((2 * n) + 1) / 24) = 64,0532.

6.3.10. Obter Zteste:

$$Z = \frac{T - \mu_T}{\sigma_T}$$

6.3.10.1. z = (T - mu_t) / sigma_t = -4.79908



6.3.11. Passo 5: Aceita ou não H0:

Teste de Wilcoxon (amostras dependentes)	
Hipóteses	H_0: não há diferença entre os grupos H_1: há diferença entre os grupos
Estatística de Teste	$Z = \frac{T-\mu_T}{\sigma_T}$
Regras de Rejeição de H_0	
Valor Crítico z	Rejeitar se $Z \leq -z_{\alpha/2}$ ou se $Z \geq z_{\alpha/2}$
Valor Crítico t	Rejeitar se $Z \leq -t_{\alpha/2}$ ou se $Z \geq t_{\alpha/2}$
Valor p	Rejeitar se $p\_value \leq \alpha$

```
z <= z_alpha_2
True
z >= z_alpha_2
False
```

6.3.12. Conclusão: Rejeitamos a hipótese de que não existe diferença entre os grupos, isto é, existe uma diferença entre as médias de cigarros fumados pelos pacientes antes e depois do tratamento. E como é possível verificar através das médias de cigarros fumados por dia antes (31.86) e depois (11.2) do tratamento, podemos concluir que o tratamento apresentou resultado satisfatório.

6.4. Critério do valor de p:

6.4.1. Utilizando a biblioteca

```
from scipy.stats import wilcoxon
```

6.4.2. Conseguimos reduzir tudo o que foi feito até agora em uma linha:

```
T, p_valor = wilcoxon(fumo.Antes, fumo.Depois)
print(T, p_valor)
```

```
True
```

```
p_valor <= significância
                                True
```

6.5. Teste de Mann-Whitney:

6.5.1. Comparação de duas populações - amostras independentes.

6.5.2. Mann-Whitney é um teste não paramétrico utilizado para verificar se duas amostras independentes foram selecionadas a partir de populações que têm a mesma média. Por ser um teste não paramétrico,

Mann-Whitney torna-se uma alternativa ao teste paramétrico de comparação de médias.

6.5.3. Problema: Em nosso dataset temos os rendimentos dos chefes de domicílio obtidos da Pesquisa Nacional por Amostra de Domicílios - PNAD no ano de 2015. Um problema bastante conhecido em nosso país diz respeito a desigualdade de renda, principalmente entre homens e mulheres. Duas amostras aleatórias, uma de 6 homens e outra com 8 mulheres, foram selecionadas em nosso dataset. Com o objetivo de comprovar tal desigualdade teste a igualdade das médias entra estas duas amostras com um nível de significância de 5%.

6.5.4. Seleção das Amostras:

6.5.4.1. mulheres = dados.query('Sexo == 1 and Renda > 0').sample(n = 8, random_state = 101).Renda;

6.5.4.2. homens = dados.query('Sexo == 0 and Renda > 0').sample(n = 6, random_state = 101).Renda;

6.5.5. Dados do Problema:

6.5.5.1. media_amostra_M = mulheres.mean() = 1090.75;

6.5.5.2. media_amostra_H = homens.mean() = 1341.66;

6.5.5.3. significancia = 0.05;

6.5.5.4. confianca = 1 – significância;

6.5.5.5. n_1 = len(homens) = 6;

6.5.5.6. n_2 = len(mulheres) = 8.

6.5.6. Passo 1: formulação das hipóteses H0 e H1

6.5.6.1. **Lembre-se, a hipótese nula sempre contém a alegação de igualdade.**

6.5.6.2. $\mu m \Rightarrow$ Média das rendas dos chefes de domicílios do sexo feminino;

6.5.6.3. $\mu h \Rightarrow$ Média das rendas dos chefes de domicílios do sexo masculino;

6.5.6.4. $H0: \mu m = \mu h$;

6.5.6.5. $H1: \mu m < \mu h$.

6.5.7. Passo 2 - escolha da distribuição amostral adequada

6.5.7.1. Deve-se optar pela distribuição t de Student, já que nada é mencionado sobre a distribuição da população, o desvio padrão

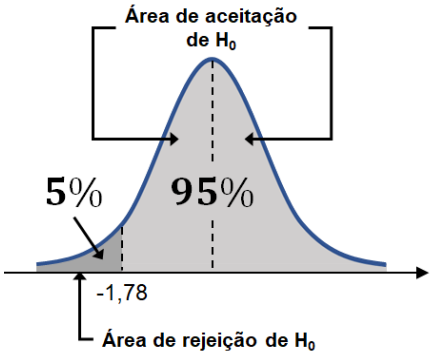
populacional é desconhecido e o número de elementos investigados é menor que 30.

6.5.8. Passo 3 - fixação da significância do teste (α):

6.5.8.1. Obtendo t_α:

6.5.8.1.1. graus_de_liberdade = n_1 + n_2 – 2 = 12;

6.5.8.1.2. t_alpha = t_student.ppf(significancia, graus_de_liberdade) = -1,78.



6.5.9. Passo 4 - cálculo da estatística-teste e verificação desse valor com as áreas de aceitação e rejeição do teste:

6.5.9.1. Definir os n's:

6.5.9.1.1. n1 = nº de elementos do menor grupo;

6.5.9.1.2. n2 = nº de elementos do maior grupo.

6.5.9.2. Obter a soma dos postos

6.5.9.2.1. R1 = soma dos postos do grupo n1;

6.5.9.2.2. R2 = soma dos postos do grupo n2.

6.5.9.3. Obter as estatísticas:

$$u_1 = n_1 \times n_2 + \frac{n_1 \times (n_1 + 1)}{2} - R_1$$
$$u_2 = n_1 \times n_2 + \frac{n_2 \times (n_2 + 1)}{2} - R_2$$

6.5.9.4. Selecionar o menor U:

$$u = min(u_1, u_2)$$

6.5.9.5. Obter a estatística de teste:

$$Z = \frac{u - \mu(u)}{\sigma(u)}$$

6.5.9.5.1. Onde:

$$\mu(u) = \frac{n_1 \times n_2}{2}$$
$$\sigma(u) = \sqrt{\frac{n_1 \times n_2 \times (n_1 + n_2 + 1)}{12}}$$

6.5.9.6. Obtendo os postos:

```
H = pd.DataFrame(homens)
H['Sexo'] = 'Homens'
M = pd.DataFrame(mulheres)
M['Sexo'] = 'Mulheres'
sexo = H.append(M)
sexo.reset_index(inplace=True, drop=True)
sexo.sort_values(by = 'Renda', inplace=True)
sexo['Posto'] = range(1, len(sexo) + 1)
posto = sexo[['Renda', 'Posto']].groupby(['Renda']).mean()
posto.reset_index(inplace=True)
sexo.drop(['Posto'], axis = 1, inplace=True)
sexo = sexo.merge(posto, left_on = 'Renda', right_on= 'Renda', how
= 'left')
sexo
```

	Renda	Sexo	Posto
0	250	Mulheres	1.0
1	400	Mulheres	2.5
2	400	Mulheres	2.5
3	700	Mulheres	4.0
4	788	Mulheres	5.5

6.5.9.7. Obtendo R:

6.5.9.7.1. R1 = soma dos postos do grupo n1;

6.5.9.7.2. R2 = soma dos postos do grupo n2;

```
temp = sexo[['Sexo', 'Posto']].groupby('Sexo').sum()
temp
```

	Posto
Sexo	
Homens	61.0
Mulheres	44.0

6.5.9.7.3. r_1 = temp.loc['Homens'][0] = 61;

6.5.9.7.4. r_2 = temp.loc['Mulheres'][0] = 44.

6.5.9.8. Obter u:

$$u_1 = n_1 \times n_2 + \frac{n_1 \times (n_1 + 1)}{2} - R_1$$
$$u_2 = n_1 \times n_2 + \frac{n_2 \times (n_2 + 1)}{2} - R_2$$
$$u = min(u_1, u_2)$$

6.5.9.8.1. u_1 = n_1 * n_2 + ((n_1 * (n_1 + 1)) / (2)) - r_1 = 8;

6.5.9.8.2. u_2 = n_2 * n_2 + ((n_2 * (n_2 + 1)) / (2)) - r_2 = 56;

6.5.9.8.3. u = min(u_1, u_2) = 8.

6.5.9.9. Obter μ(u):

$$\mu(u) = \frac{n_1 \times n_2}{2}$$

6.5.9.9.1. mu_u = (n_1 * n_2) / 2 = 24.

6.5.9.10. Obter σ(u):

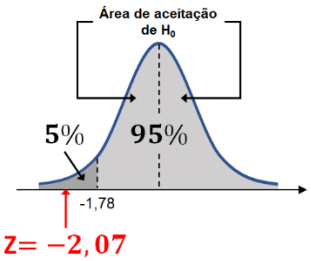
$$\sigma(u) = \sqrt{\frac{n_1 \times n_2 \times (n_1 + n_2 + 1)}{12}}$$

6.5.9.11. sigma_u = np.sqrt((n_1 * n_2 * (n_1 + n_2 + 1)) / 12) = 7.745966.

6.5.10. Obter Z:

$$Z = \frac{u - \mu(u)}{\sigma(u)}$$

6.5.10.1. z = (u - mu_u) / sigma_u = -2.07



6.5.11. Passo 5 - Aceitação ou rejeição da hipótese nula:

	Teste da Cauda Inferior	Teste da Cauda Superior	Teste Bicaudal
Hipóteses	$H_0: \mu_1 = \mu_2$ $H_1: \mu_1 < \mu_2$	$H_0: \mu_1 = \mu_2$ $H_1: \mu_1 > \mu_2$	$H_0: \mu_1 = \mu_2$ $H_1: \mu_1 \neq \mu_2$
Estatística de Teste	$Z = \frac{u - \mu(u)}{\sigma(u)}$		
Regras de Rejeição de H_0			
Valor Crítico z	Rejeitar se $Z \leq -z_\alpha$	Rejeitar se $Z \geq z_\alpha$	Rejeitar se $Z \leq -z_{\alpha/2}$ ou se $Z \geq z_{\alpha/2}$
Valor Crítico t	Rejeitar se $Z \leq -t_\alpha$	Rejeitar se $Z \geq t_\alpha$	Rejeitar se $Z \leq -t_{\alpha/2}$ ou se $Z \geq t_{\alpha/2}$
Valor p		Rejeitar se $p\_value \leq \alpha$	

6.5.11.1. Rejeitar H_0 se $Z \leq -t_\alpha$:

```
z <= t_alpha
```

```
True
```

6.5.12. Conclusão: Rejeitamos a hipótese de que não existe diferença entre os grupos, isto é, concluímos que a média das rendas dos chefes de domicílios do sexo feminino é menor que a média das rendas dos chefes de domicílios do sexo masculino. Confirmando a alegação de desigualdade de renda entre os sexos.

6.6. Aplicando p-valor:

6.6.1. Importando lib:

```
from scipy.stats import mannwhitneyu
```

6.6.2. Resolvendo tudo em uma linha:

```
u, p_valor = mannwhitneyu(mulheres, homens, alternative='less')
print(u, p_valor)
```

```
8.0 0.022221119551528605
```

```
p_valor <= significância
```

```
True
```

6.6.3. Ou seja, mesma conclusão que acima resumido em 1 linha.

6.7. O que aprendemos:

6.7.1. A aplicação de testes não paramétricos de comparação entre amostras dependentes;

6.7.2. A elaboração do teste não paramétrico de Wilcoxon;

6.7.3. A aplicação de testes não paramétricos de comparação entre amostras independentes;

6.7.4. A elaboração do teste não paramétrico de Mann-Whitney;

6.7.5. A utilização do ferramental Python para aplicação dos testes de Wilcoxon e Mann-Whitney.

7. Aula 7 – Resumo e Projeto Final:

7.1. O que aprendemos:

7.1.1. A fixação do entendimento sobre testes de hipóteses paramétricos e não paramétricos;

7.1.2. A aplicação das técnicas desenvolvidas no treinamento para solução de problemas reais.