Estatística

▼ Numérico:

É o tipo mais comum de dados;

Representa diversos tipos de coisa que você pode estar medindo

> Ex: Peso de pessoas, tempo de carregamento de página, preço de estoque,

Basicamente, coisas que pode mensurar, coisas que tem um grande potencial de possibilidades,

▼ Tipos de dados numericos:

▼ Discreto:

Baseado em inteiros.

Pode ser contado em uma série de eventos

Ex: Quantas compras um cliente realizou em um ano?

> Ele não comprou 2.25 coisas, é um inteiro, uma compra, duas compras..

Quantas vezes eu virei minha cabeça?

Basicamente o discreto impede valores que não sejam de fato inteiros.

▼ Continuo:

Tem infinitos valores possiveis.

Ex: Uma pessoa poderia ter desde 5 pés de altura até 10,37625 polegadas de altura ou vice versa

O tempo que levo para realizar o check out em um site

Existe uma gama de possibilidades.

Havendo uma possibilidade de quantidades infinitamente grandes, onde há possibilidade de precisão neles.

▼ Categórico:

É o tipo de de data sem significado numérico inerente, você não consegue comparar uma categoria com outra diretamente.

Ex: Genero(Yes/No) VS Race (Branco, Preto, Amarelo, Pardo,...)

Você pode atribuir números a estas categorias, porém esses numero não irão se interar devido a natureza desses dados ser diferente entre si.

É possivel através dos dados, dizer que Texas é maior que a Florida, porém, não da para simplismente dizer que o Texas é maior.

É possivel clasificar porém, de diversos modos de uma atribuição numerica para cada estado.

Eu posso dizer que a flórida é o estado numero 3 e Texas numero 4, sem que haja um relacionamento entre estas duas informações, basicamente é uma abreviação, como um ID para cada um afim de classifica-los.

Logo, Categorizar informações, é basicamente dividir valores instrincicos. É um modo de separar valores em conjuntos de dados pela categoria

▼ Ordinal:

É uma mistura dos dados categóricos com os numéricos

Ex: Estrelas em uma classificação de filmes

Existe uma classificação de 5 valores, dentre eles, que 1 é ruim e 5 é bom.

Mas não existe matematicamente um significado matemático para isto

Deste modo, conseguimos mensurar niveis de um determinado valor

▼ Mean(media)

Somatório de todos valores de uma lista, pela contagem desta lista.

Ex: contei nove placas, das quais existiam valores de 0 a 2. O somatório das placas é igual a =10, logo, a média dos valores destas placas é 1.11 por placa

▼ Median(Mediana)

Pego esses valores de placas, é encontrado o valor que se encontra na metade. Logo, se nas placas estavam:

2,0,1,2,0,1,2,0,1 - Contagem

0,0,1,1,1,1,2,2,2 - Separação

0,0,1,1,(_1_),1,2,2,2 - Encontro da mediana

Caso a quantidade de valores impossibilite encontrar a mediana, realizamos a media entre os 2 valores centrais, deste modo o resultado é igual a mediana.

0,0,1,1,1,1,1,2,2,2

1+1/2=1 (Mediana)

Outro exemplo é que a renda familiar média nos estados unidos é de 72.641, porém, a mediana é de 51.939 devido haver uma distorção neste valor devido os bilionários.
Logo é possivel identificar através dos 2 diferenças que possibilitam interpretar se os dados são válidos para analise ou não.

▼ Mode(Moda):

É o resultado da maior repetição de dados em um dataset. Basicamente o valor mais comum em um DataSet.

2,0,1,2,0,1,2,0,1,1 - Contagem

Mode = 1

▼ Jupyter

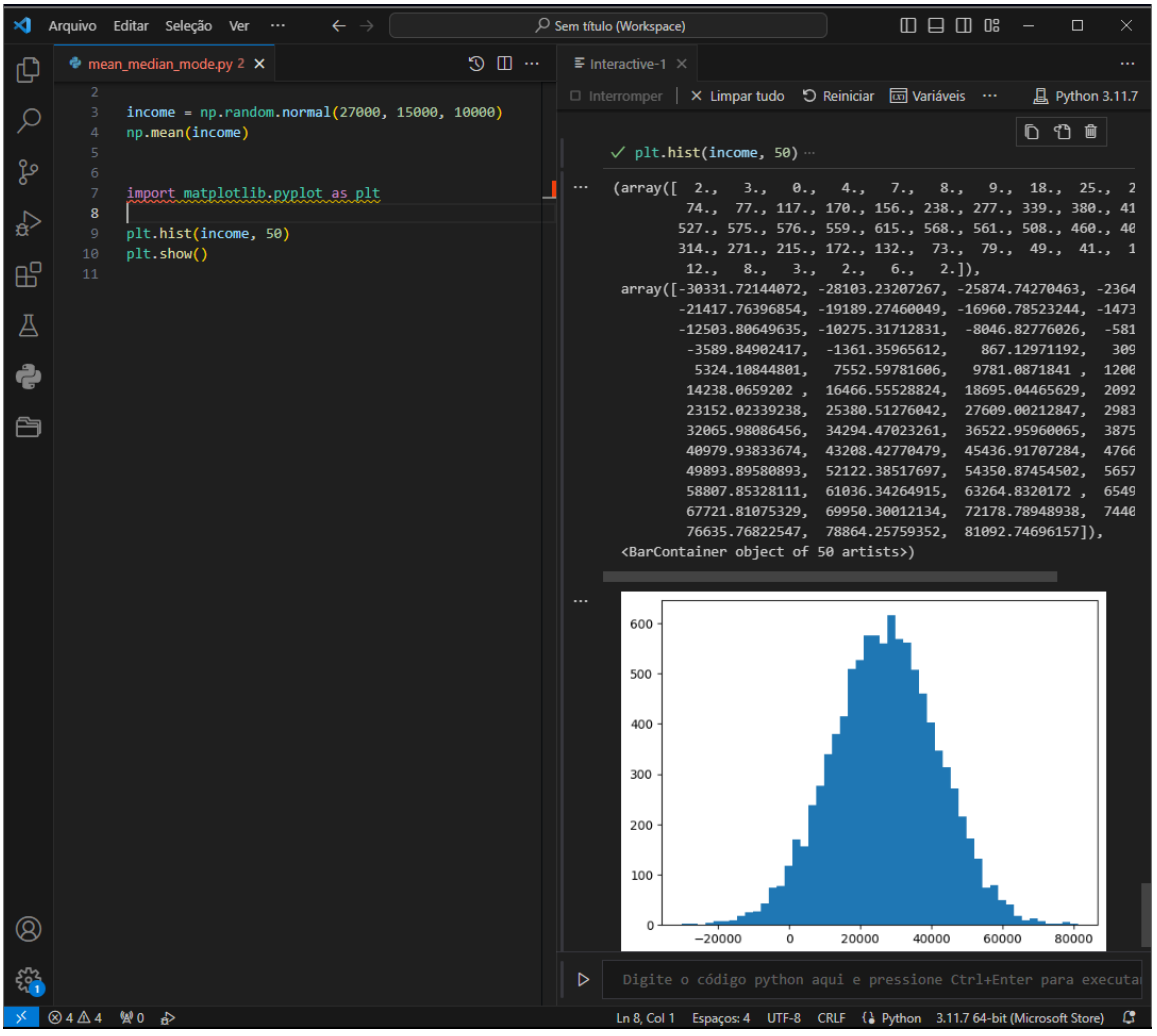
Ao utilizar o plotli, deverá ser especificado no inicio(Ao utilizar o jupyter) a seguinte linha de código:

```
%matplotlib inline
```

Caso não seja indicada, poderá gerar erros ou não constar gráfico em questão

Comando para geração de um gráfico histograma, do qual retorna a distribuição de uma amostra de dados, é utilizado seguinte código:

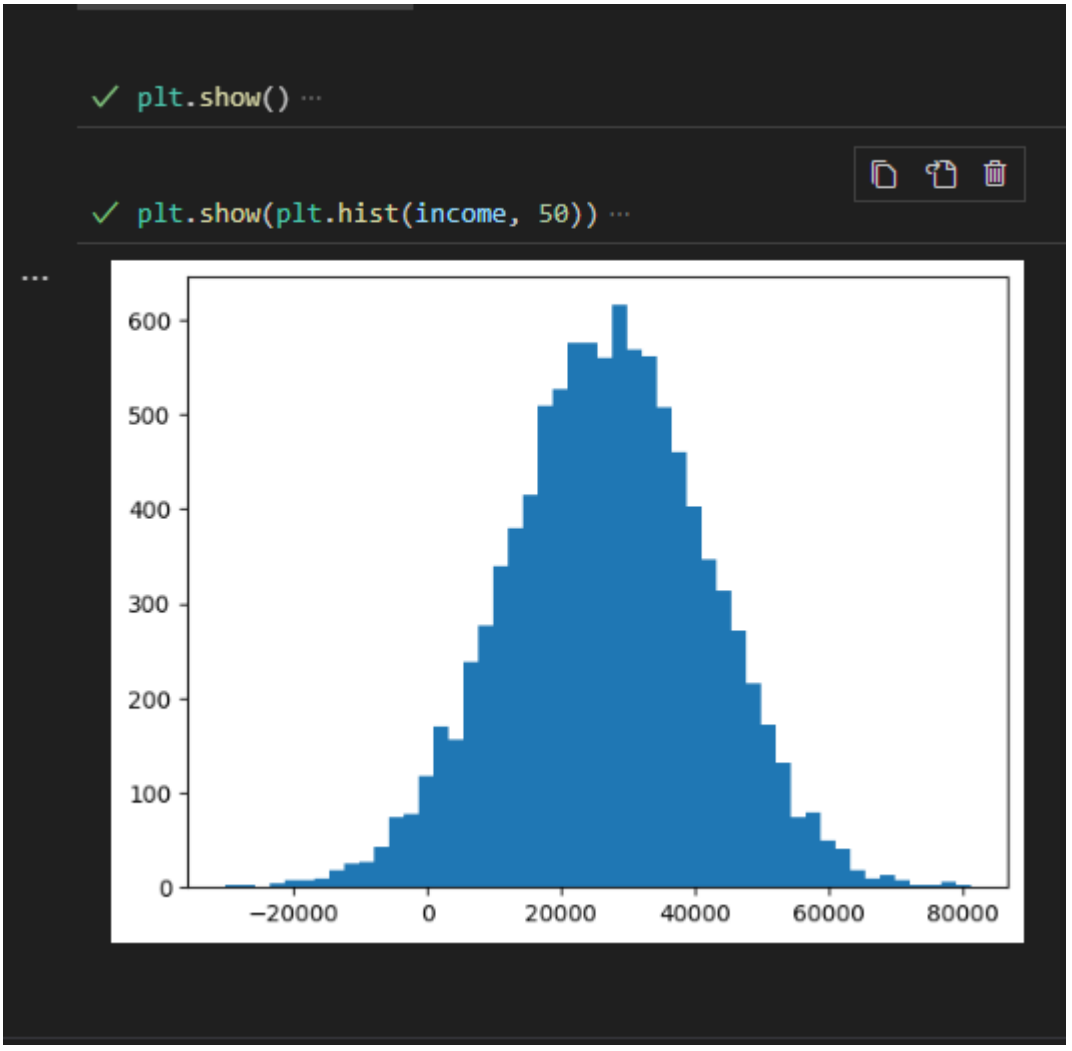
```
import matplotlib.pyplot as plt
plt.hist(income, 50)
# hist para geração deste modelo.
# quanto maior o valor indicado após variavel, maior será 
plt.show()
```



Ao utilizar o plt.show() ele não retornou nada, muito possivelmente pelo uso no vs. Para “ajustar” utilizei seguinte código:

```
plt.show(plt.hist(income, 50))
```

Deste modo retorno foi somente o gráfico, não indicando o array de definição dele.



Para esta distribuição em questão, os valores ficaram distribuídos de maneira “igualitária” de modo, que a média e a mediana representam valores muito próximos

▼ Atividade

-Now, find the mean and median of this data. In the code block below, write your code, and see if your result makes sense:

```
np.mean(incomes)
np.median(incomes)

incomes.max()
len(incomes)
```

Primeiro gerei a media. Ela retornou:

100.0806510054167

Após, gerei mediana:

100.09715771078191

Deste modo, pude ter um parametro do qual os 2 valores se encontram muito próximos. Realizei processo de validação de qual o menor valor:

13.699659114374015

E do maior valor:

183.86850296064557

Após isto, confirmei se quantidade estava batendo com geração.

10000

A partir disso, o resultado fez sentido, já que o lançamento de 10.000 valores distribuídos entre 13 - 183 poderiam gerar a media entre 100 assim como mediana também.

▼ Variação

Como determinado grupo de dados, varia entre si em um agrupamento.

Como exemplo, pode se usar os gráficos gerados no tópico jupyter, do qual no ponto médio, existe uma maior quantidade de vezes que determinados valores são próximos. Logo, variação, fala sobre como se encontra o seu agrupamento de dados.

Também dita como média das diferenças em relação à média

DataSet:

1, 4, 5, 4, 8 = Média = 4.4

4.4-1=-3.4

4.4-4=-0.4

4.4-5=0.6

4.4-4=-0.4

4.4-8=3.6

Após isto, elevado ao quadrado:

11.56, 0.16, 0.36, 0.16, 12.96

e após isto, fazemos novamente a média entre estes valores.

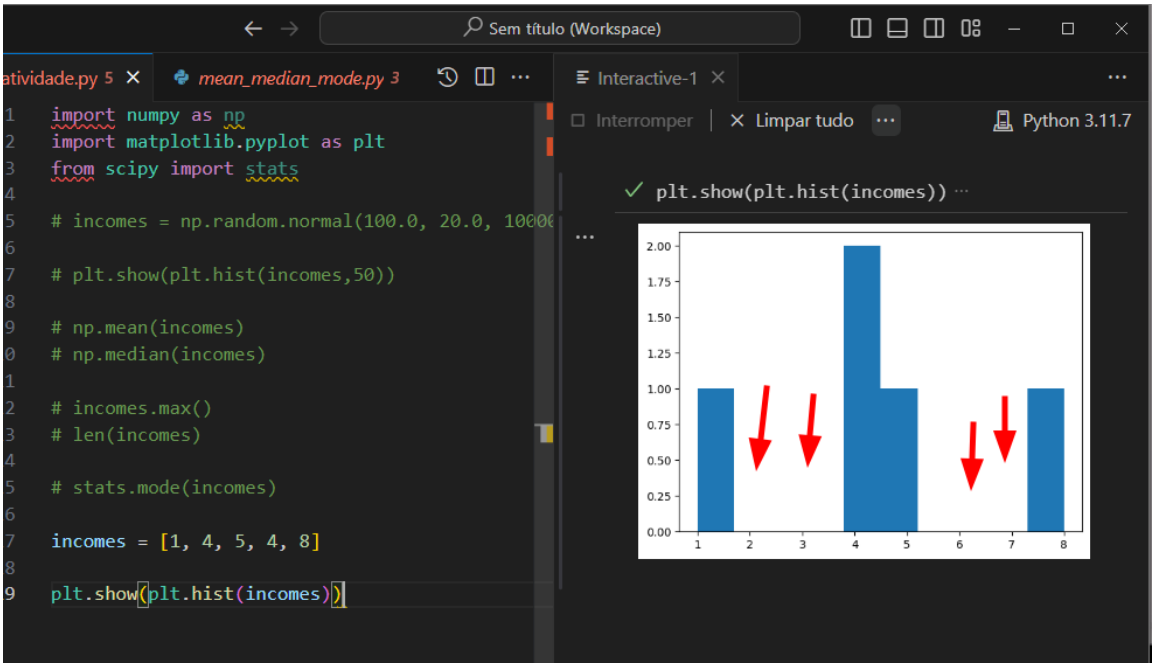
Neste caso, queremos que todos os valores fiquem positivos, por isso elevamos ao quadrado, também queremos que haja maior peso nos valores atípicos

Ao fim, temos o valor 5.04.

▼ Desvio

Para encontrar o desvio, utilizamos a raiz da variação.

É utilizada para encontrar o caminho de identificação dos valores atipicos na base de dados



Estando os valores 1 e 8 distantes da média, podemos falar que o valor 1 e 8 são valores atípicos, devido estarem fora do desvio, que neste exemplo é 2.24

▼ População Vs Variância da amostra

Ao realizar processo de conferencia da variação de DataSets muito grandes, é utilizando processo de Sample, que se difere somente na quantidade que é dividido os elementos na média final. Logo, se contagem é igual a 5 elementos, ao realizar a média final é indicado um N-1 sob o valor da divisão, alterando assim resultado final.

Porém, se você detém da base de dados completa, você realizará sem subtração, pegando valor atual dos elementos

▼ Função de densidade de probabilidade

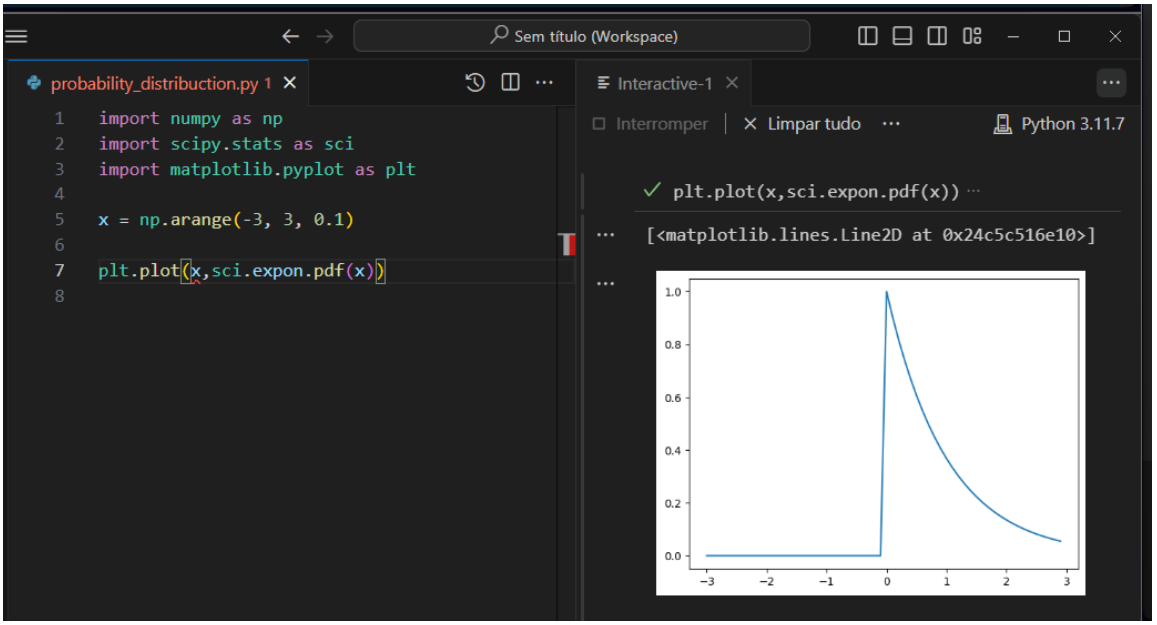
Os valores referentes a determinado valor ocorrer.

Da a probabilidade de um data point se situar dentro de um determinado intervalo de valores.

Função de densidade de probabilidade:

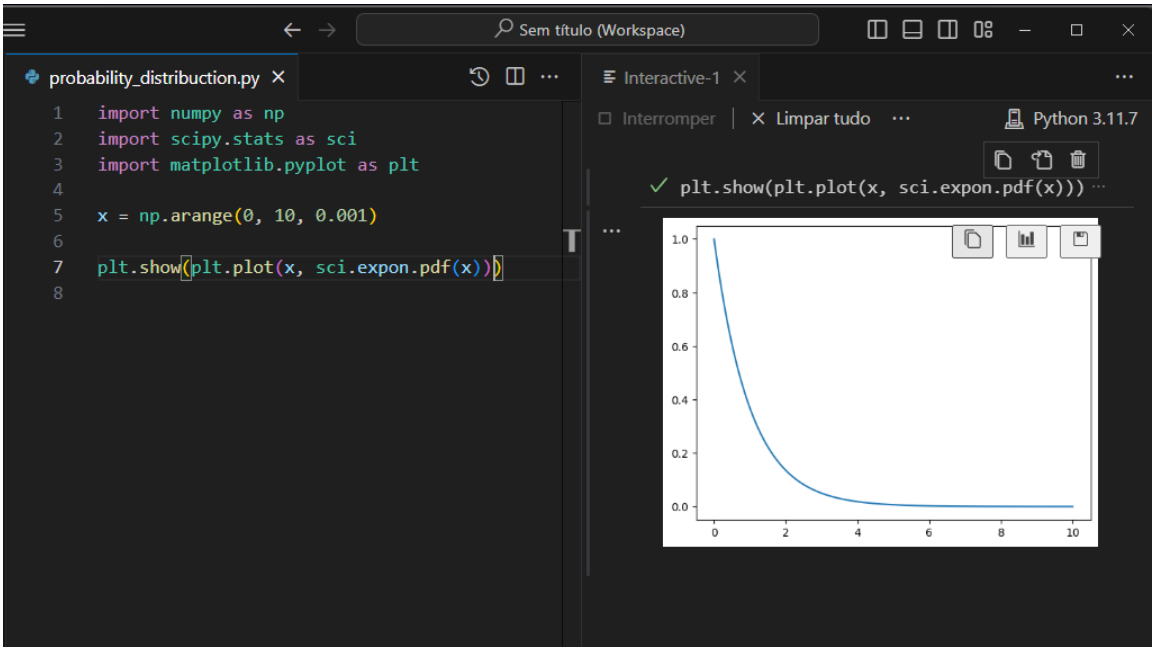

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

x = np.arange(-3, 3, 0.1)

plt.plot(x,norm.pdf(x))
```

Função exponencial:




```
import numpy as np
import scipy.stats as sci
import matplotlib.pyplot as plt

x = np.arange(-3, 3, 0.1)

plt.plot(x,sci.expon.pdf(x))
```

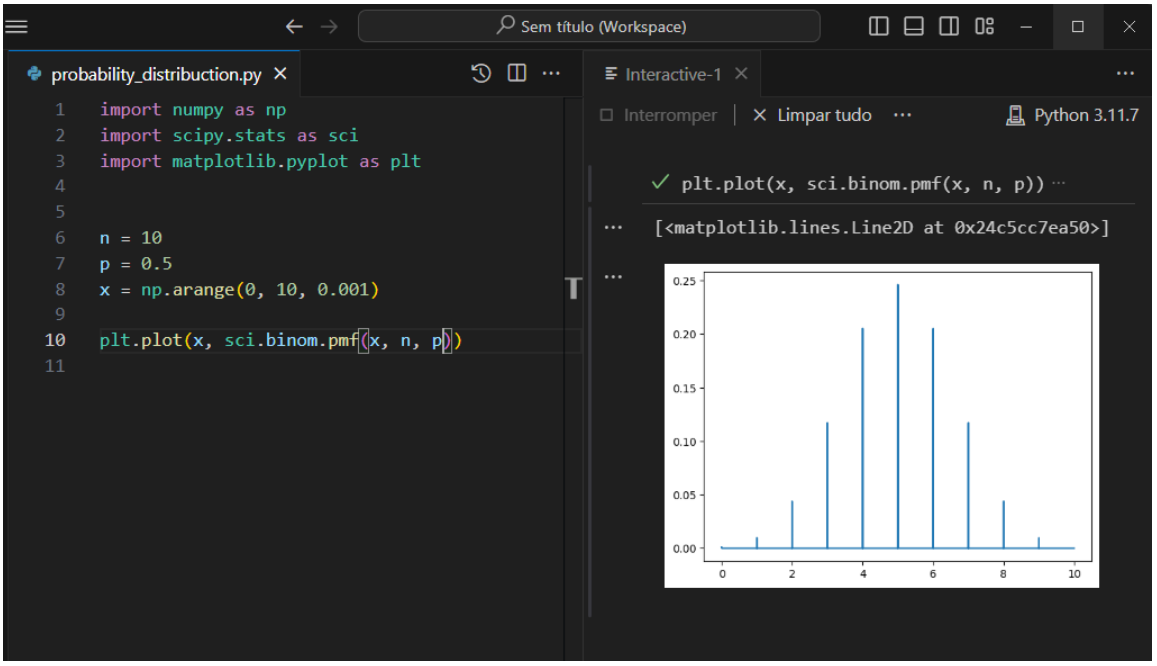
Outro exemplo:




```
import numpy as np
import scipy.stats as sci
import matplotlib.pyplot as plt

x = np.arange(0, 10, 0.001)

plt.show(plt.plot(x, sci.expon.pdf(x)))
```

Função de massa binominal:

Definimos o comprimento = N e a altura = P, após isto, definimos o conjunto de numeros, e plotamos ele com função binominal:



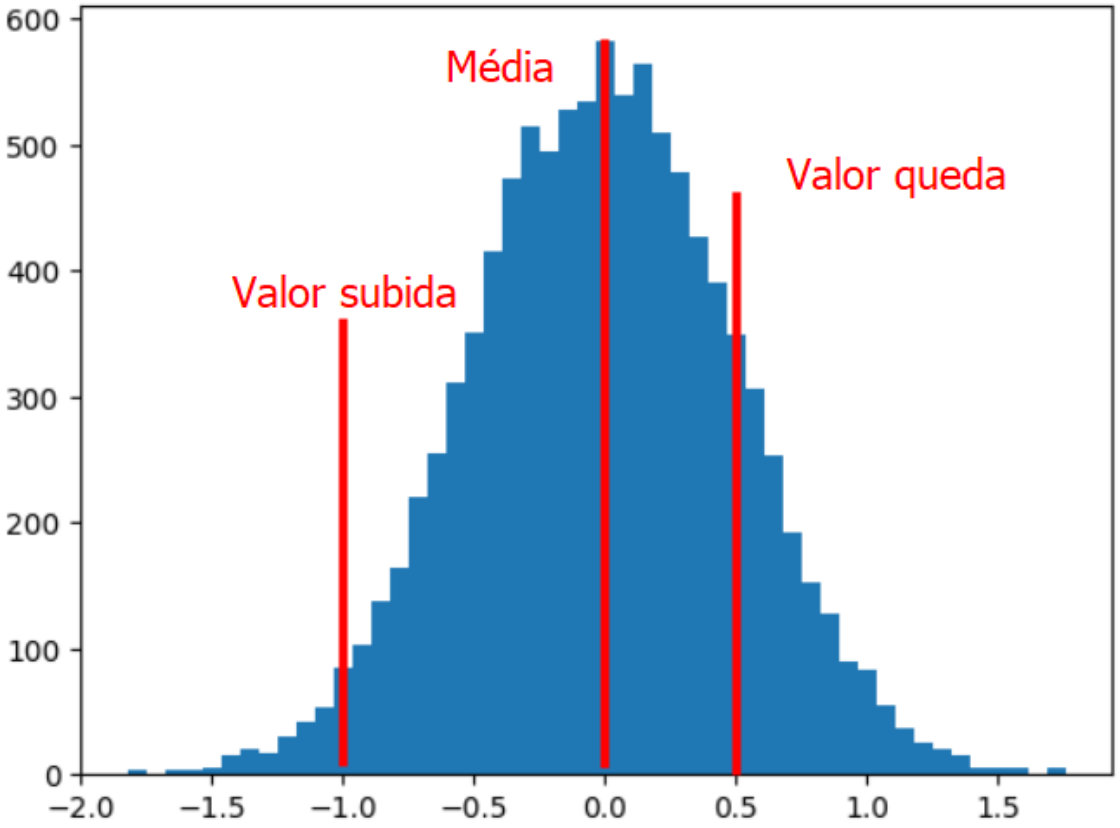

```
import numpy as np
import scipy.stats as sci
import matplotlib.pyplot as plt


n = 10
p = 0.5
x = np.arange(0, 10, 0.001)

plt.plot(x, sci.binom.pmf(x, n, p))
```

Distribuição de Poisson (Poisson distribution):

É uma distribuição de probabilidade discreta que expressa a probabilidade de um determinado número de eventos ocorrer em um intervalo fixo de tempo ou espaço se esses eventos ocorrerem com uma taxa média constante conhecida e independentemente do tempo desde o último evento

▼ Percentual

Através da função np.percentile é possivel identificar o local do qual existe maior percentual de “perda” definido pela queda no gráfico:



Se jogarmos a função np.percentile(x,50)

Temos o percentual referente a queda ou subida, neste caso, 50% ainda esta havendo uma subida de valor, mesmo sendo a média

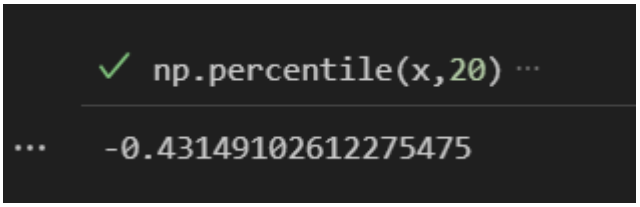
```
✓ np.percentile(x,50) ···
···  -0.007788801785918219
```

Já se alterarmos parâmetro para (x,90) encontramos seguinte valor:

```
✓ np.percentile(x,90) ···
···  0.6272926199944157
```

Do qual percentual de queda se encontra em 62% do gráfico, este que tem 90% de queda já.

Já se colocarmos valor (x, 20) encontramos o valor de subida deste gráfico, encontrado nesse esquema como %:



▼ Momento

O primeiro momento, é definido na média.

O segundo é a variancia

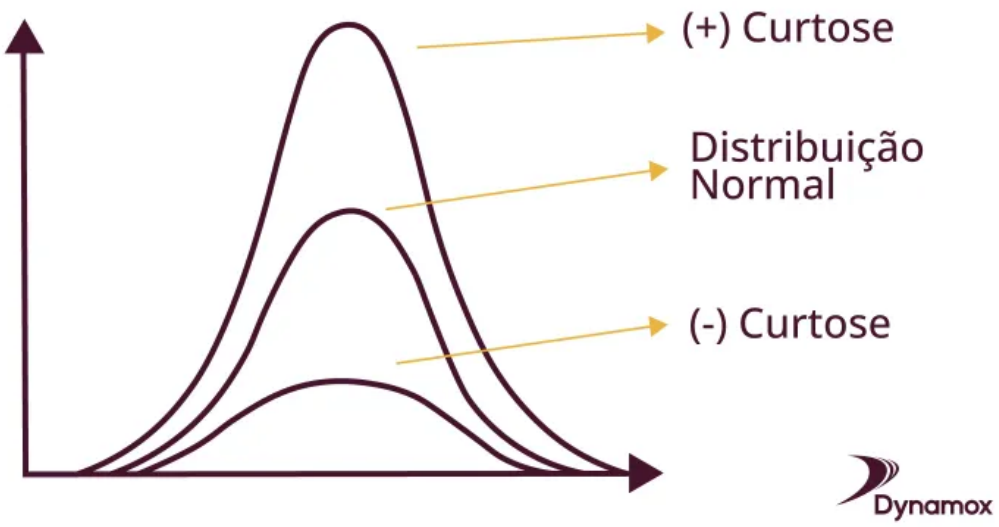
O terceiro é a inclinação

O quão desigual esta a distribuição

Basicamente a inclinação que percorre o maior caminho, como exemplo, se pegar a função de potencia, temos um valor total de Skew, afinal, ela é basicamente formada por ele, só que de ponta “cabeça”

O quarto momento é chamado de curtose

Em poucas palavras, é o quão achatada uma curva de função é:



Ela mede o grau de pico dos dados

Para calcularmos estes valores através do python, utilizamos biblioteca Scipy.

Realizando importação, podemos utilizar tanto o

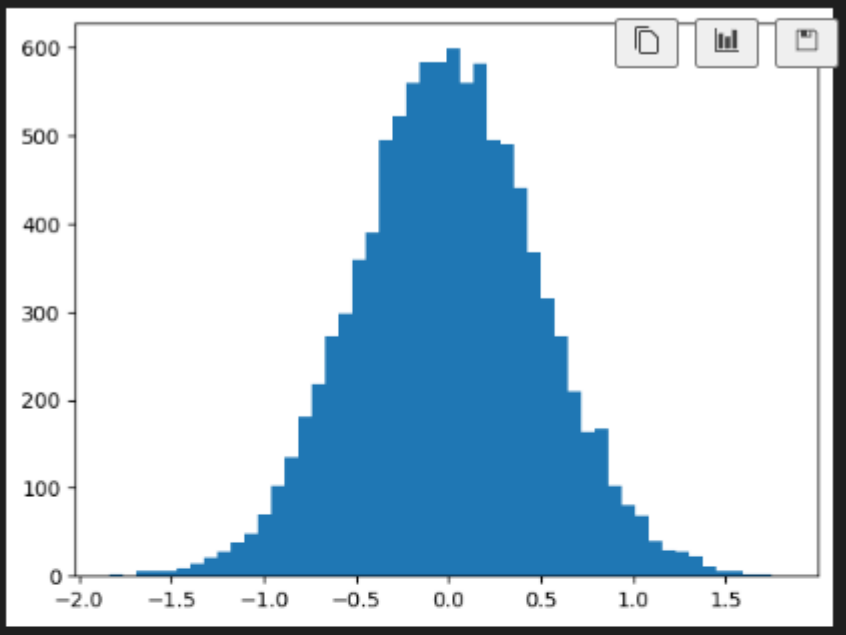
scipy.stats.skew(dataset);

scipy.stats.kurtosis(dataset)

Deste modo, ao utilizarmos as formulas aprendidas anteriormente, conseguimos encontrar o momento:

Interromper | Limpar tudo | Reiniciar ··· Python 3.11.8

```
plt.show(plt.hist(vals, 50)) ···
```




```
np.mean(vals) ···
-0.0007842605786558875

np.var(vals) ···
0.24505674201607017

sp.kurtosis(vals) ···
0.07123183249331966

sp.skew(vals) ···
0.02039090479139978
```

▼ MatPlotLib

É uma biblioteca para geração de gráficos.

Podemos utilizar alguns comandos através desta biblioteca, permitindo personalizar ou demonstrar nossos gráficos:

Plot:

Comando para gerar o gráfico, depende do show para mostra-lo.

Show:

Comando para demonstrar o gráfico plotado.

Savefig:

Comando para gravar seu gráfico em um arquivo.

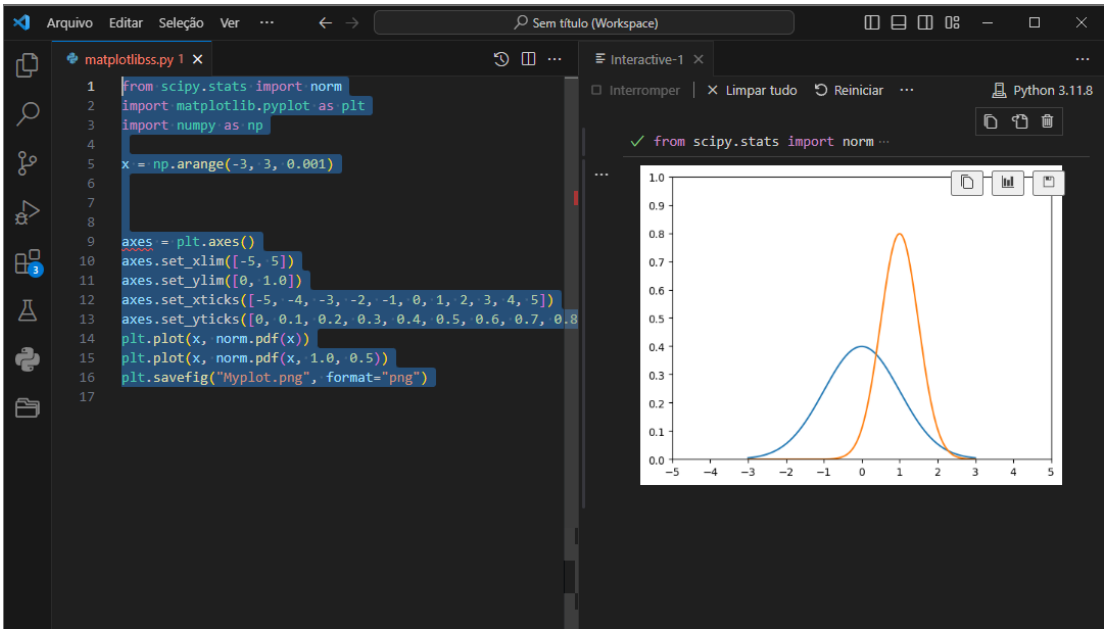
Axes:

Permite definir quais serão os índices X e Y que seu gráfico irá seguir.

- axes.set_xlim - Indice X
  - axes.set_xticks - valores que irão constar neste índice X
- axes.set_ylim - Índice Y
  - axes.set_yticks - Valores que irão constar no índice Y

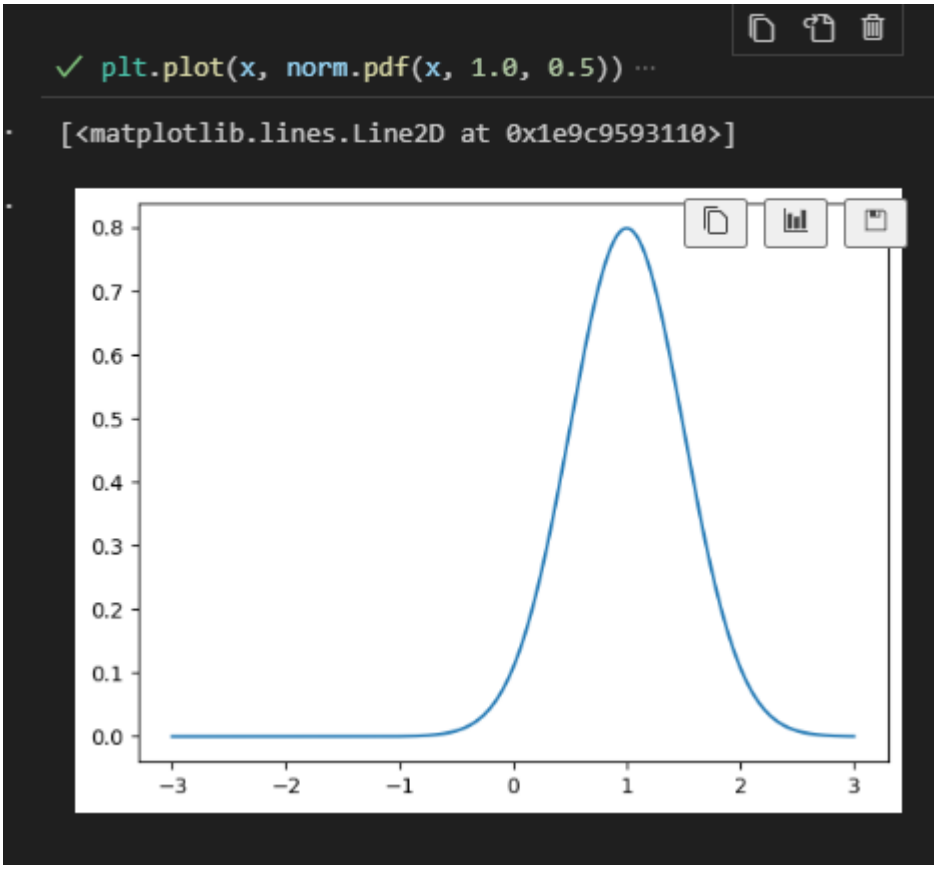
Exemplo:

```
axes = plt.axes()
axes.set_xlim([-5, 5])
axes.set_ylim([0, 1.0])
axes.set_xticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])
axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0
plt.plot(x, norm.pdf(x))
plt.plot(x, norm.pdf(x, 1.0, 0.5))
```

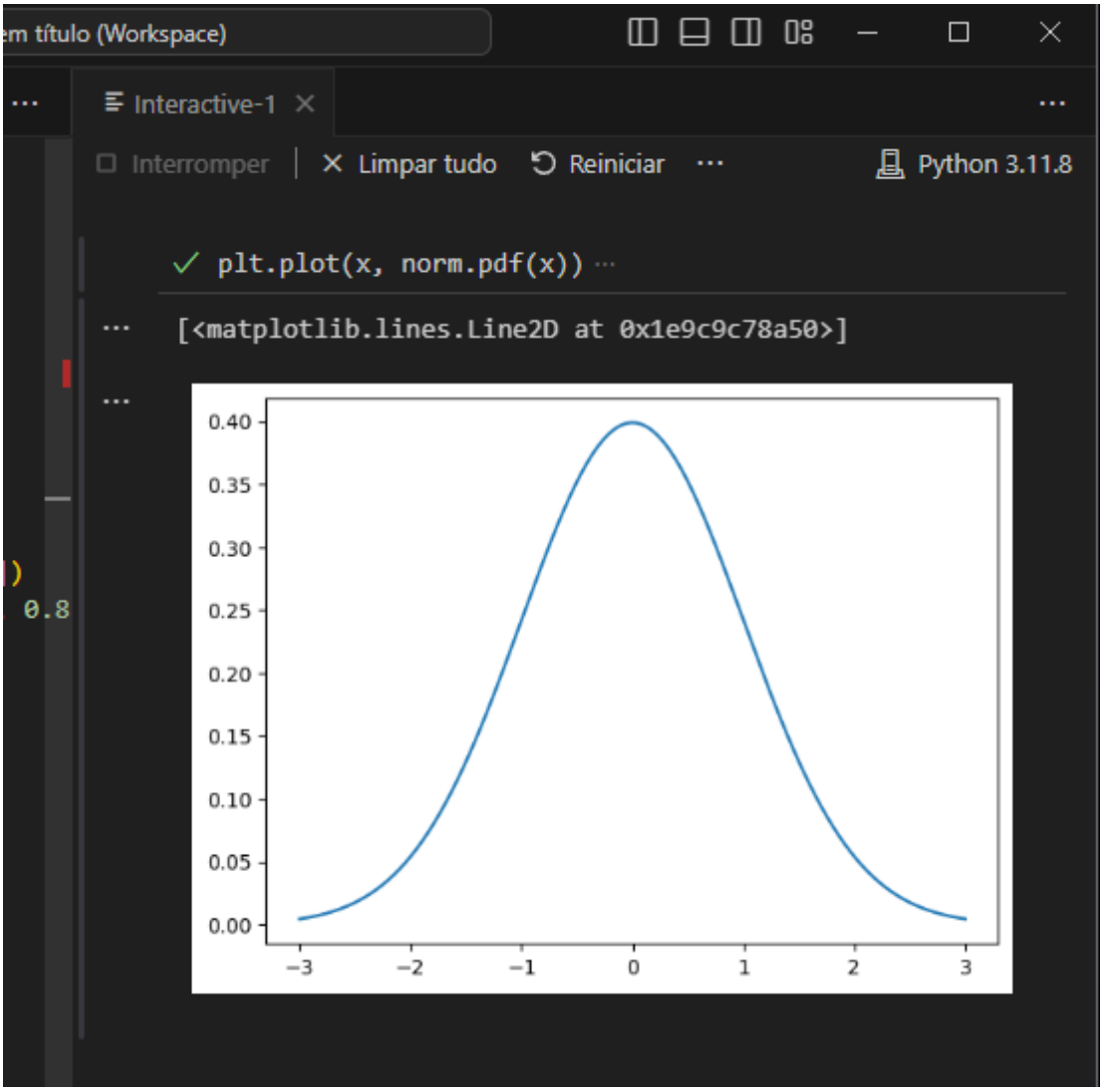


Para este caso em questão, é gerado um array, através da biblioteca numpy e então utilizado ele para geração de 2 gráficos, um que é definido manualmente o índice X através da complementação do parâmetro norm.pdf e o outro que trás somente um índice “normal” com array x:

```
plt.plot(x, norm.pdf(x, 1.0, 0.5))
```




```
plt.plot(x, norm.pdf(x))
```



Outras funções:

Axes.grid:

Adiciona um grid no gráfico(Caderno de matemática)

plt.xlabel:

Adiciona um nome para índice X

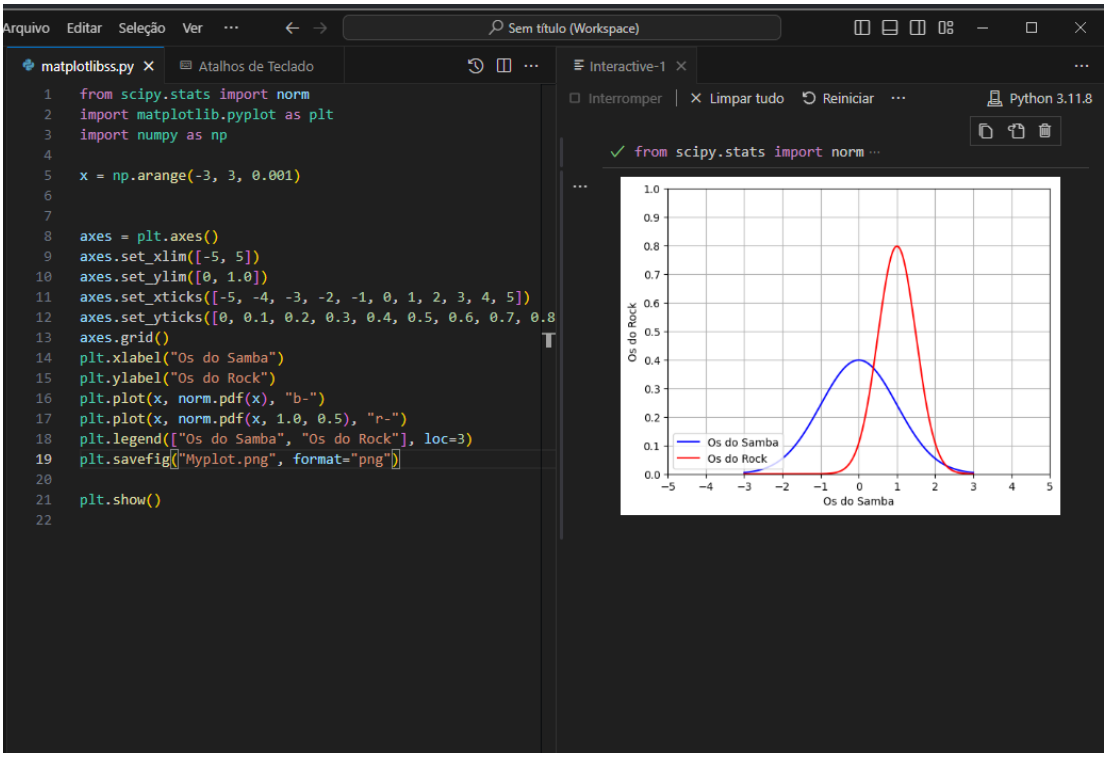
plt.ylabel:

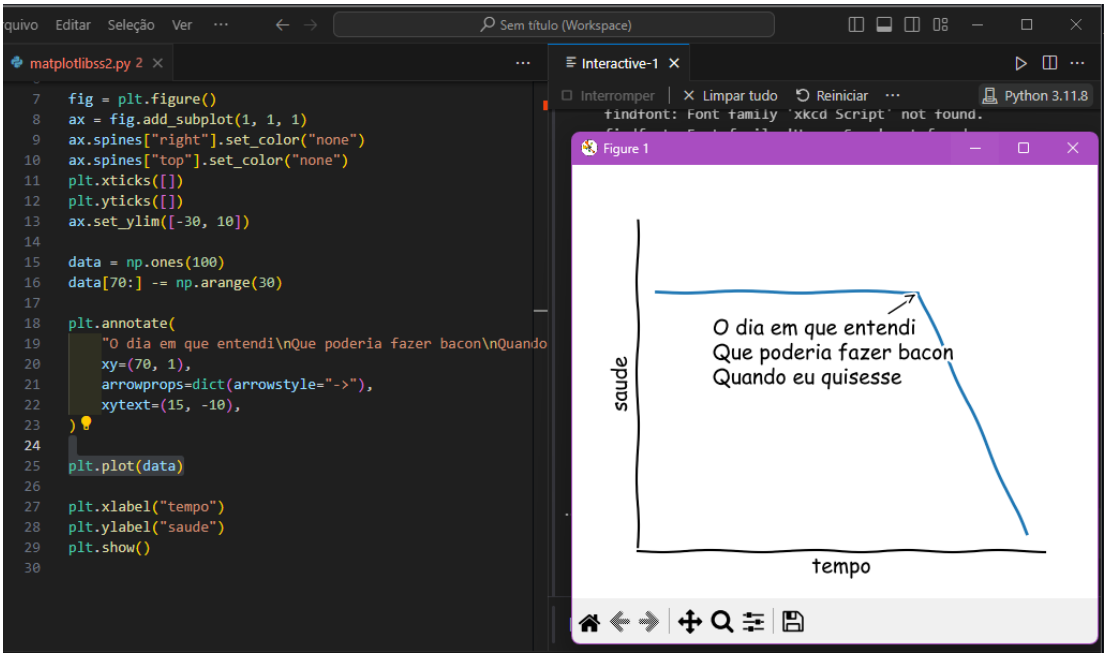
Adiciona um nome para índice Y

plt.legend

Adiciona uma legenda que pode ou não utilizar os labels

```
axes = plt.axes()
axes.set_xlim([-5, 5])
axes.set_ylim([0, 1.0])
axes.set_xticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])
axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0
axes.grid()
plt.xlabel("Os do Samba")
plt.ylabel("Os do Rock")
plt.plot(x, norm.pdf(x), "b-")
plt.plot(x, norm.pdf(x, 1.0, 0.5), "r-")
plt.legend(["Os do Samba", "Os do Rock"], loc=3)
plt.savefig("Myplot.png", format="png")
```





Na linha

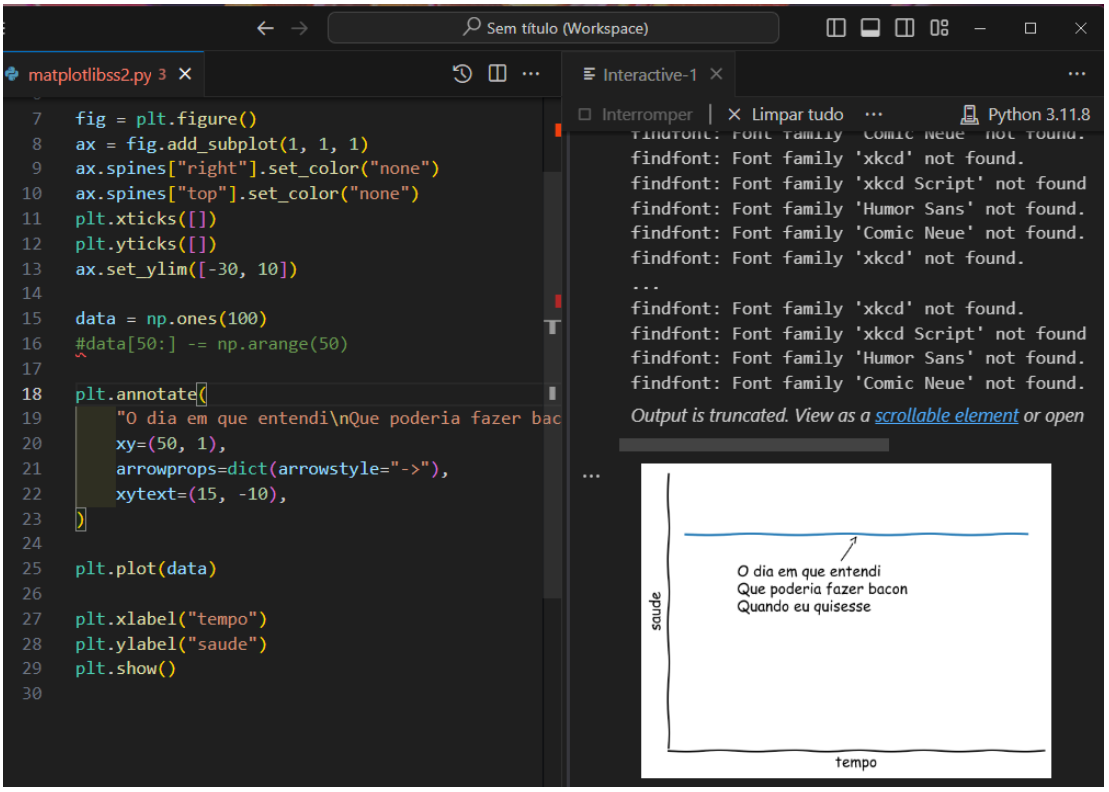
```
plt.annotate(
    "O dia em que entendi\nQue poderia fazer bacon\nQuar
    xy=(57, -5),
    arrowprops=dict(arrowstyle="->"),
    xytext=(15, -10),
)
```

“xy = (57, -5)” esta linha controlará a flexa referente a imagem. Podendo ela ter um parâmetro positivo ou negativo no seu eixo Y referido ali por -5.

Outro fato é o uso de um aray para subtrair valores, utilizados com data[50:] -= np.arange(50)

Valor precisa ser exato para gerar 100, caso contrário irá gerar critica, deste modo, parâmetro irá trabalhar com um escalar de perda, que é a queda do gráfico

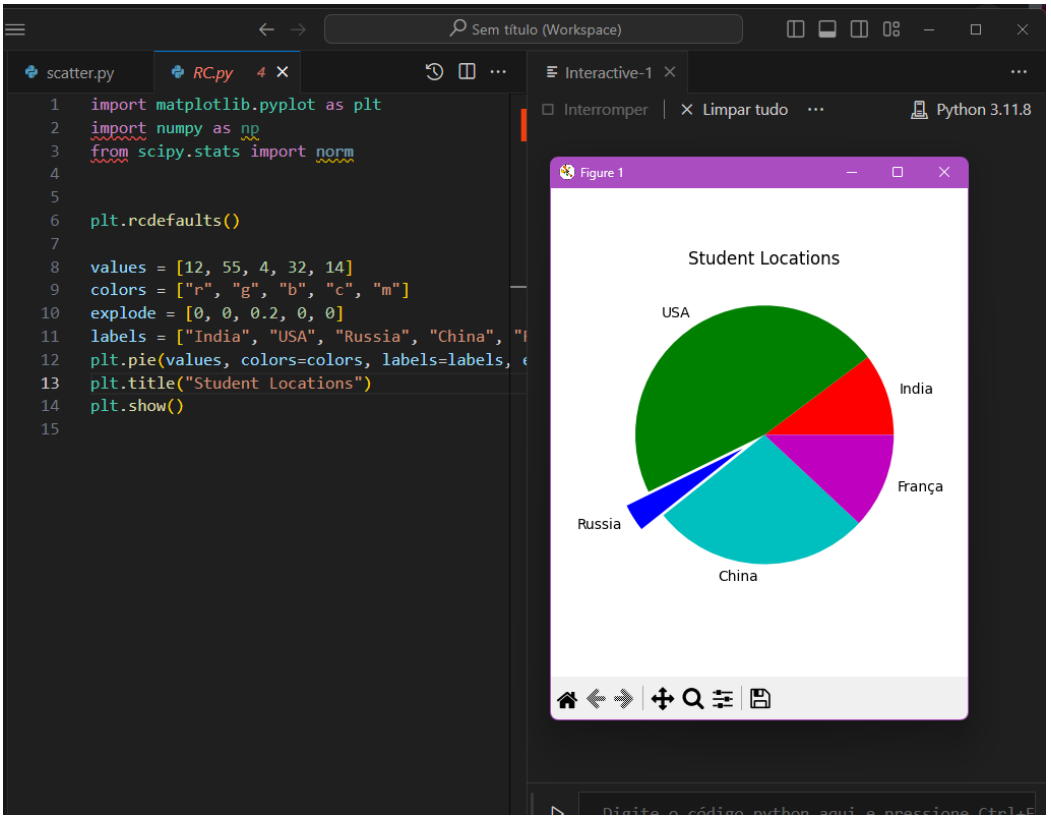
```
data = np.ones(100)
data[50:] -= np.arange(50)
```



Retirando termo, retorno é uma linha reta.

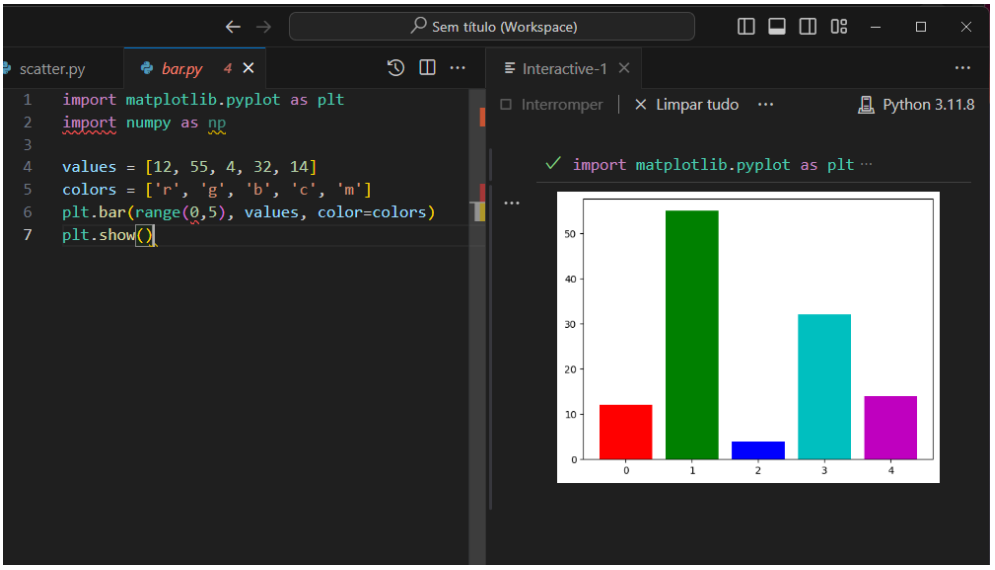
- RC(PIZZA ou PIE)

  Gerado a partir do comando plt.pie, indicar parâmetro ‘Explode’, para que haja um indicador externo exibindo o valor desejado:



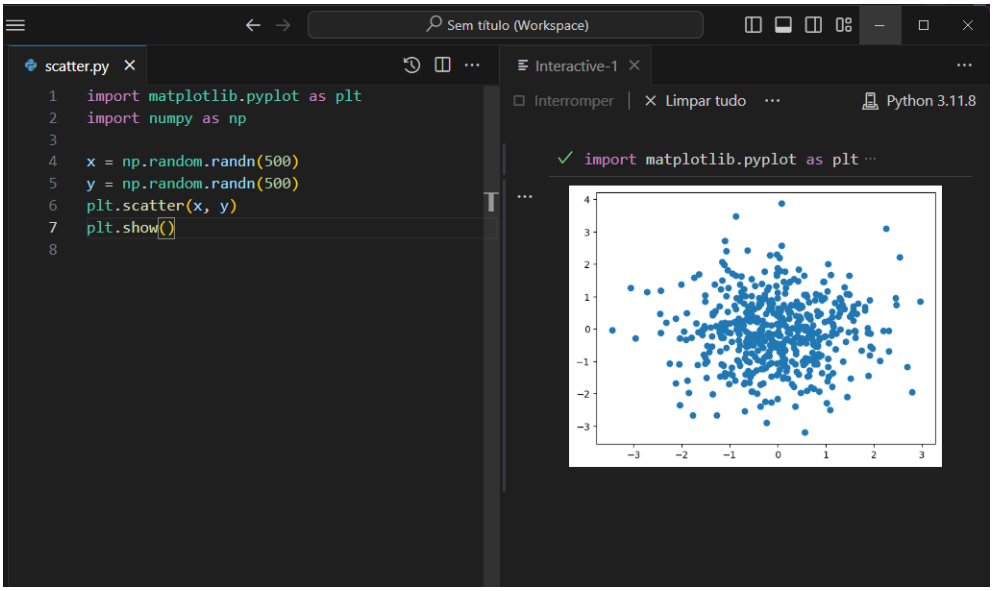

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm


plt.rcdefaults()

values = [12, 55, 4, 32, 14]
colors = ["r", "g", "b", "c", "m"]
explode = [0, 0, 0.2, 0, 0]
labels = ["India", "USA", "Russia", "China", "França"
plt.pie(values, colors=colors, labels=labels, explode
plt.title("Student Locations")
plt.show()
```

- Bar(barra)
  - Gera um gráfico de barras comum:



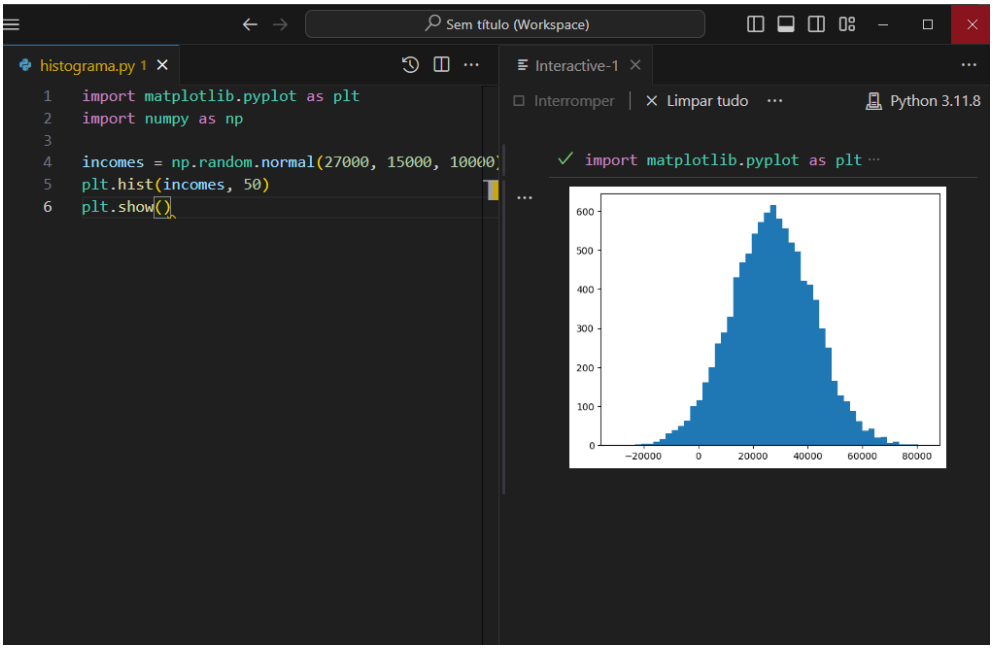

```
import matplotlib.pyplot as plt
import numpy as np

values = [12, 55, 4, 32, 14]
colors = ['r', 'g', 'b', 'c', 'm']
plt.bar(range(0,5), values, color=colors)
plt.show()
```

- Scatter
  - É possível definir ponto X e Y do qual gráfico indicará esses pontos dinamicamente em um quadro:



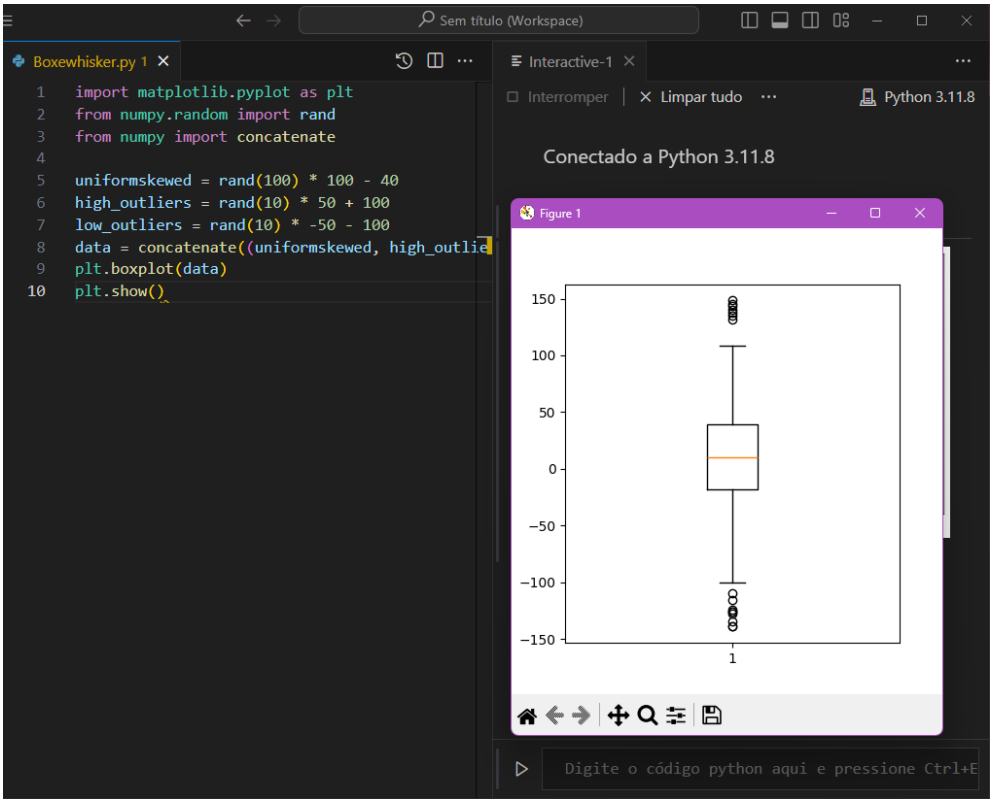
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randn(500)
y = np.random.randn(500)
plt.scatter(x, y)
plt.show()
```

- Histograma
  - Tipo de gráfico padrão, do qual, indica um crescimento de pico, semelhante a uma montanha:




```
import matplotlib.pyplot as plt
import numpy as np

incomes = np.random.normal(27000, 15000, 10000)
plt.hist(incomes, 50)
plt.show()
```

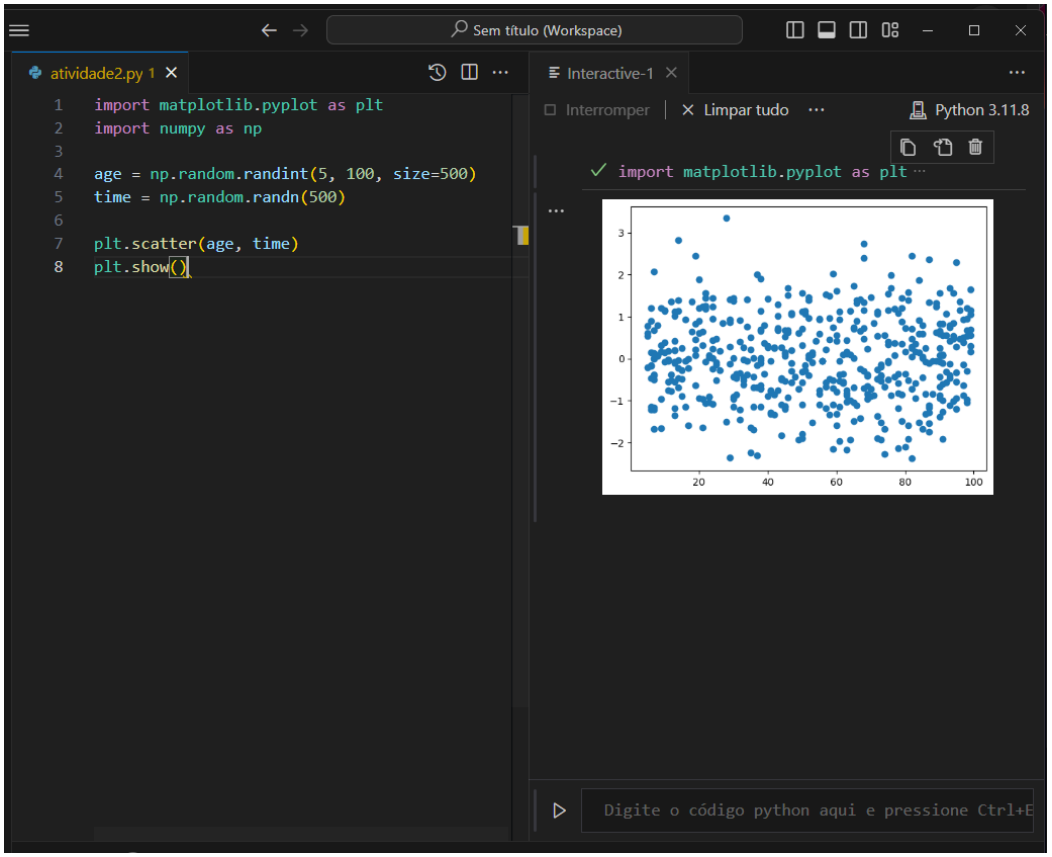
- Box & Whiskers(Quartis)
  - Define os quartis e indica os valores deles




```
import matplotlib.pyplot as plt
from numpy.random import rand
from numpy import concatenate

uniformskewed = rand(100) * 100 - 40
high_outliers = rand(10) * 50 + 100
low_outliers = rand(10) * -50 - 100
data = concatenate((uniformskewed, high_outliers,
plt.boxplot(data)
plt.show()
```

- Atividade:

- Foi indicado para gerar através do modelo scatter, um gráfico que representasse idade por tempo assistindo TV:




```
import matplotlib.pyplot as plt
import numpy as np

age = np.random.randint(5, 100, size=500)
time = np.random.randn(500)

plt.scatter(age, time)
plt.show()
```

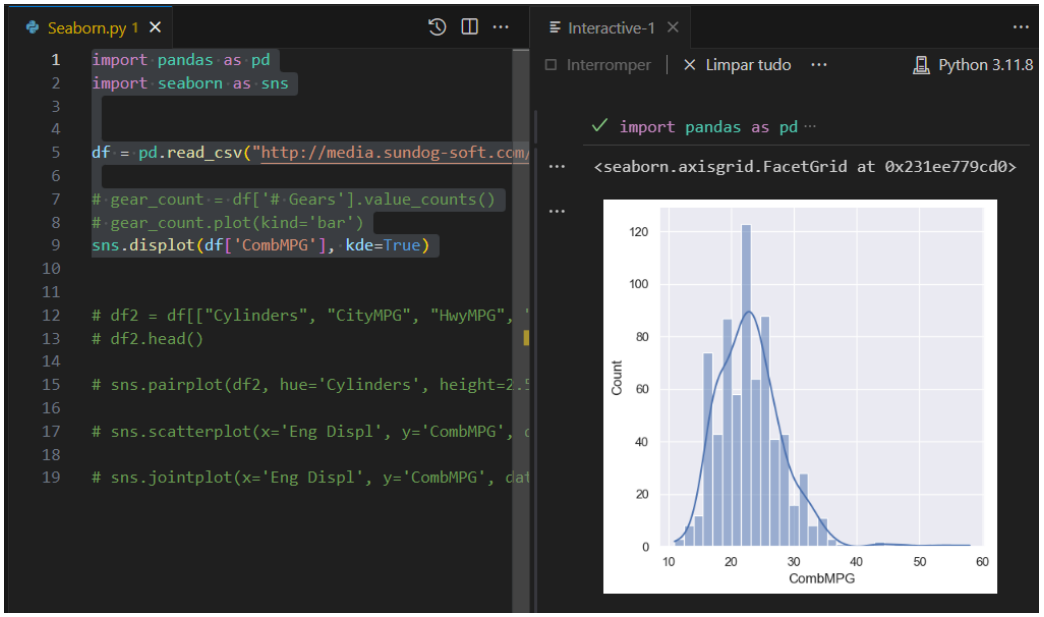
▼ Seaborn

Basicamente um matplotlib++. Permite mais funções, aparencia fica mais bonita e da outras capacidades extras.

- Features:

▼ Displot

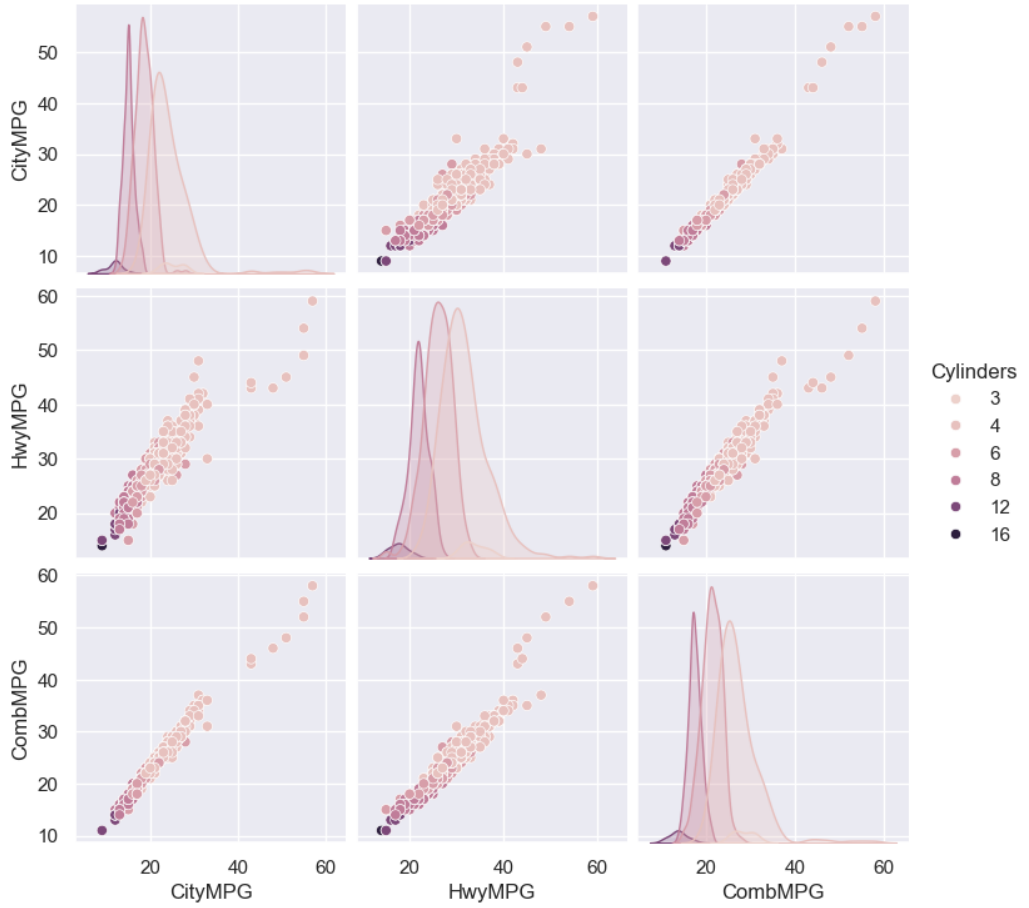
Permite visualizar um histograma, porém utilizando função KDE é possivel fazer que o mesmo apresente uma curva referente a um gráfico normalizado:




```
import pandas as pd
import seaborn as sns


df = pd.read_csv("http://media.sundog-soft.com/SelfD

# gear_count = df['# Gears'].value_counts()
# gear_count.plot(kind='bar')
sns.displot(df['CombMPG'], kde=True)
```

▼ Pairplot

É possivel juntar, através de um dataframe, diversas informações e cruzalas com um parametro que elas compartilham, como exemplo, foi utilizado a quantidade de cilindros em um carro para medir médias de consumo de combustivel:




```
df2 = df[["Cylinders", "CityMPG", "HwyMPG", "CombMPG
df2.head()

sns.pairplot(df2, hue='Cylinders', height=2.5 )
```

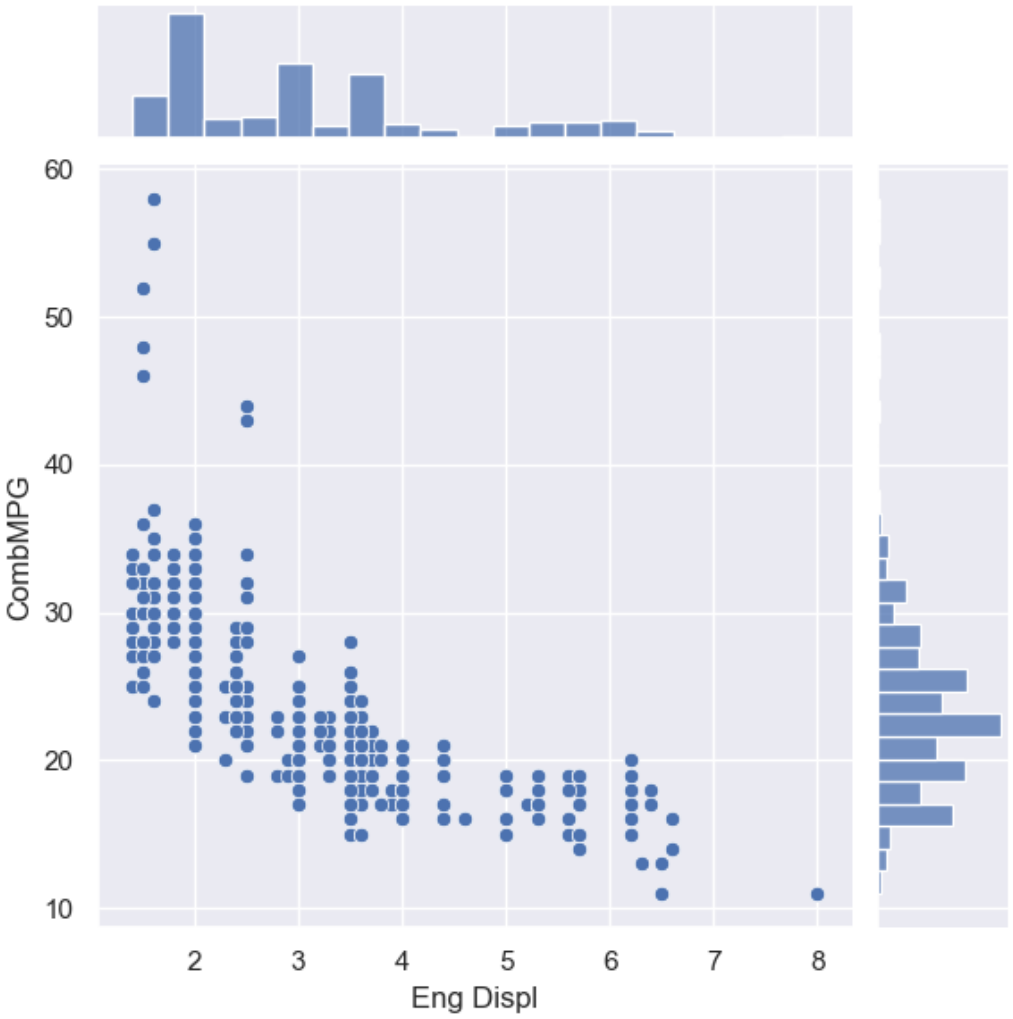
▼ ScatterPlot

Identico com o apresentado no matplotlib:


```
sns.scatterplot(x='Eng Displ', y='CombMPG', data=df)
```

▼ Joinplot

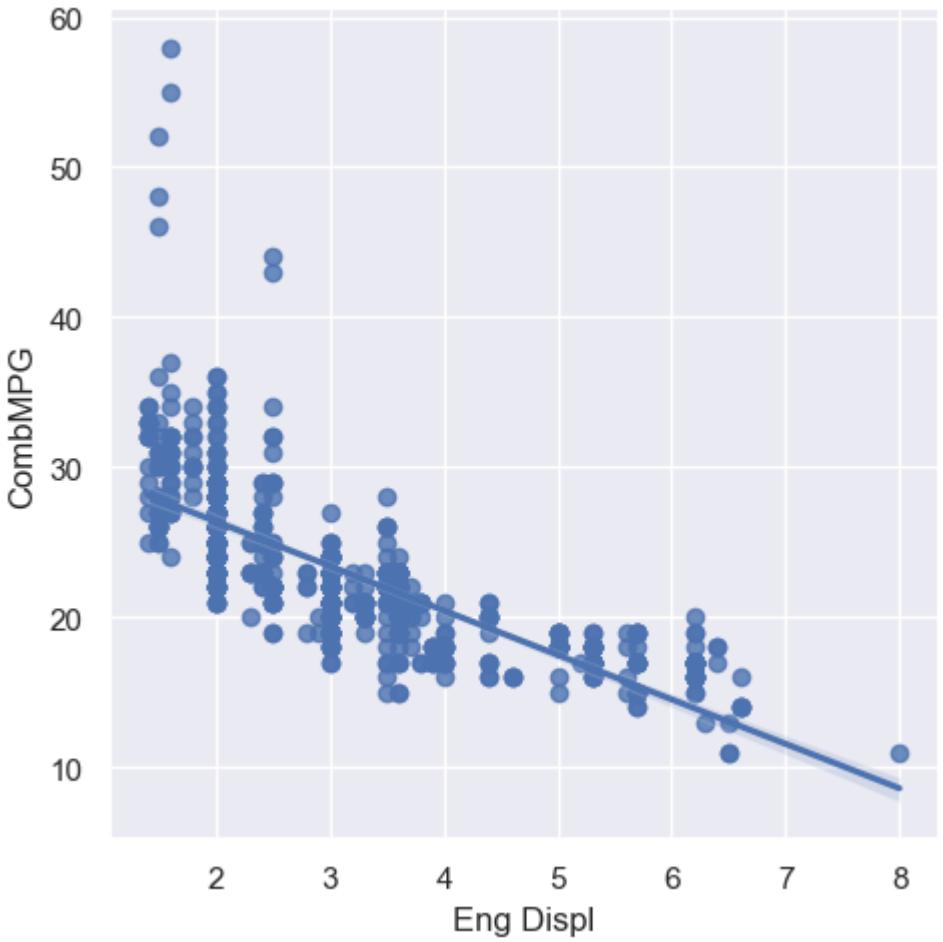
Une o scatterPlot porém com um gráfico externo para medir quantidade de repetições de um ponto:




```
sns.jointplot(x='Eng Displ', y='CombMPG', data=df)
```

▼ Lmplot

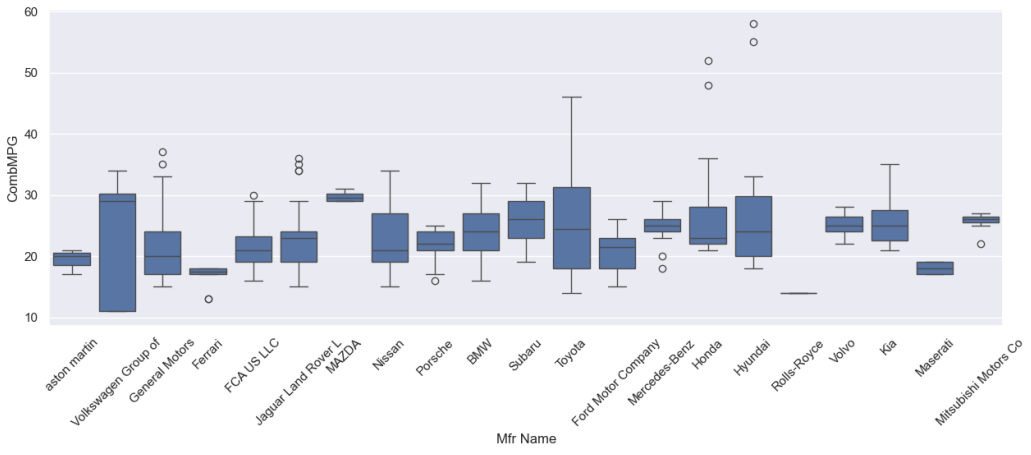
É o scatterPlot com uma linha definindo uma regressão linear:




```
sns.lmplot(x='Eng Displ', y='CombMPG', data=df)
```

▼ Boxplot

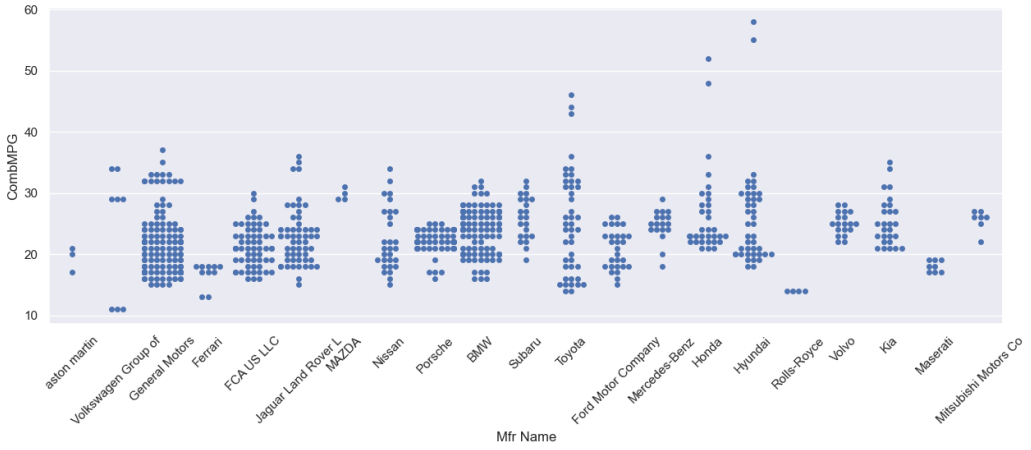
Permite gerar um gráfico de caixa:



```
sns.set_theme(rc={"figure.figsize": (15, 5)})
ax = sns.boxplot(x="Mfr Name", y="CombMPG", data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45
```

▼ Swarmplot

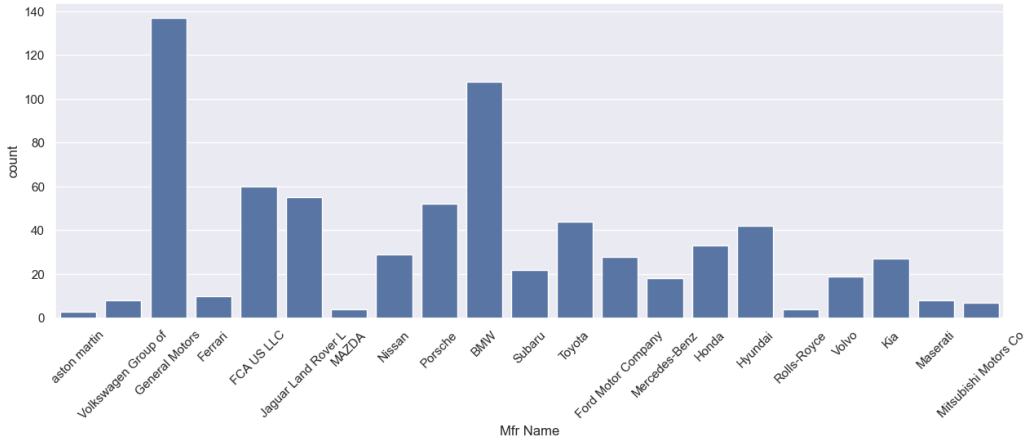
Semelhante ao boxplot, porém, define quantidade através de pontos:




```
ax = sns.swarmplot(x="Mfr Name", y="CombMPG", data=d
ax.set_xticklabels(ax.get_xticklabels(), rotation=45
```

▼ CountPlot

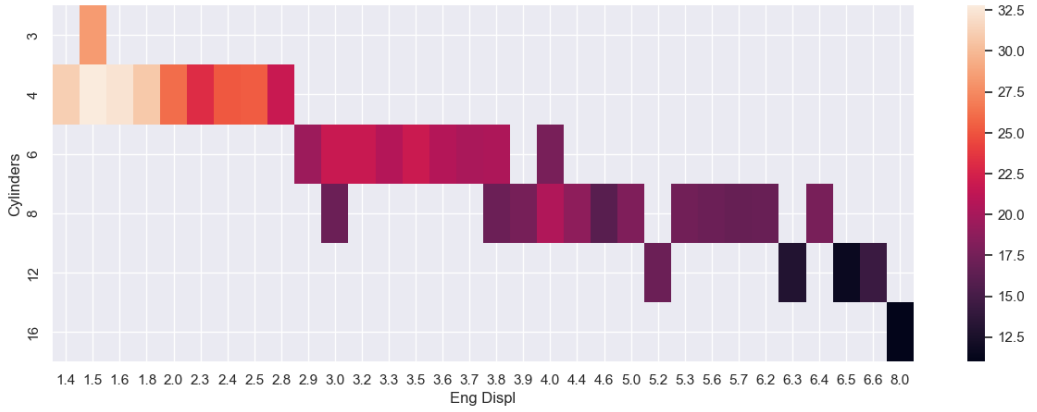
Gera um gráfico de barras, baseado na quantidade de itens de uma determinada coluna:



```
ax = sns.countplot(x="Mfr Name", data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45
```

▼ Pivotagem

Permite uma demonstração de 3 informações em um unico gráfico 2d que representa um indice através de cores “quentes”:




```
df2 = df.pivot_table(
    index="Cylinders", columns="Eng Displ", values="
)
sns.heatmap(df2)
```

Neste caso, permite identificar qual a média combinada de Highway e cidade que os carros fazem, sendo escuro como menor e claro para maior.

▼ Covariancia

É a média das diferenças combinadas. Havendo duas variáveis e as duas tem retorno zero a partir da forma de covariância, as mesmas são independentes

▼ Correlação

Determina o quanto 2 atributos se relacionam. É utilizado o coeficiente de correlação para o cálculo da função de covariância

> “Portanto, uma **covariância** positiva sempre resulta em uma **correlação** positiva e uma **covariância** negativa sempre resulta em uma **correlação** negativa.”

▼ Probabilidade Condicional

Processo de encontrar a porcentagem de determinada coisa ocorrer, baseada numa formula.

Ex:

Tenho 5 pedras de bingo em minha bolsa. 3 são vermelhas e 2 azuis.

A probabilidade de eu pegar uma azul é de 2/5, mas caso eu pegue uma azul, como ficarão as probabilidades de pegar outra azul?

Resposta 1/5 ou 1/4?

Logicamente 1/4, havendo assim uma diferença a cada tentativa.

Durante as aulas foram direcionados alguns exemplos, dos quais, não ficaram muito claros. Fato é, probabilidade condicional, dará a probabilidade possivel, mediante os valores em jogo, dos quais, diminuem a partir do momento que são chamados.

```
from numpy import random

random.seed(0)

totals = {20: 0, 30: 0, 40: 0, 50: 0, 60: 0, 70: 0}
purchases = {20: 0, 30: 0, 40: 0, 50: 0, 60: 0, 70: 0}

# totals[20] -=1
# print(totals)

totalPurchases = 0

for _ in range(100000):
    ageDecade = random.choice([20, 30, 40, 50, 60, 70])
    purchaseProbability = float(ageDecade) / 100.0
    totals[ageDecade] += 1
```

```
    if random.random() < purchaseProbability:
        totalPurchases += 1
        purchases[ageDecade] += 1


PEF = float(purchases[30]) / float(totals[30])
print(f'P(purchase - 30s) : {str(PEF)}')
totals
PF = float(totals[30]) / 100000.0
print(f"P(30s) : {str(PF)}")

PE = float(totalPurchases) / 100000.0
print(f'P(Purchase) {str(PE)}:')
```

Para isto, foi utilizado 2 formulas, uma que a partir de uma probabilidade, chama os valores entre 20 - 70, após isto, faz um calculo da possibilidade do mesmo gerar ou não uma “venda”

Após isto, foi indicado o calculo de P(E), P(F), PEF.

▼ Bayes’ Theorem

A probabilidade de A-B é a probabilidade de A vezes a probabilidade de B-A dividido pela probabilidade de B

É a dependencia tanto da probabilidade de A quanto da probabilidade da B. Em outras palavras, ele proporciona, que consigamos distinguir que um usuário de drogas que dá positivo num teste é diferente de uma pessoa que deu positivo para um teste de drogas.

Logo, nem toda pessoa que deu um positivo no teste será de fato usuário e o teorema esta aqui para identificar essa diferença.

Sendo assim, digamos que 0.3% da população usa uma droga, sendo 99% das vezes possível de identificar um usuário de drogas com teste mas 99% mas existe uma acurácia de 99% das vezes serem também não usuários.

Sendo assim:

P(B) é 1.3%(0.99∗0.003 + 0.01 ∗ 0.997) Probabilidade do teste ser positivo se você usar * a probabilidade de dar positivo se você não usar.

Sendo resposta de 22.8%. Logo as chances de alguém que é usuário de drogas, testar e dar positivo é somente de 22.8%.

Toda vez que o P(B-A) for alto não significa que o P(A-B) será alto também.