# Matplotlib Python

June 12, 2017

## 1 Visualização de dados em Python com Matplotlib

**1. Introdução** - Matplotlib é provavelmente o pacote Python mais utilizado para gráficos 2D. Ele fornece uma maneira muito rápida de visualizar dados de Python e números de qualidade de publicação em vários formatos. - O Matplotlib vem com um conjunto de configurações padrão que permitem personalizar todos os tipos de propriedades. Você pode controlar os padrões de quase todas as propriedades em matplotlib: tamanho de figura e dpi, largura de linha, cor e estilo, eixos, propriedades de eixo e grade, texto e propriedades de fonte e assim por diante. Enquanto os padrões de matplotlib são bastante bons na maioria dos casos, você pode querer modificar algumas propriedades para casos específicos.

**Exemplo 1**

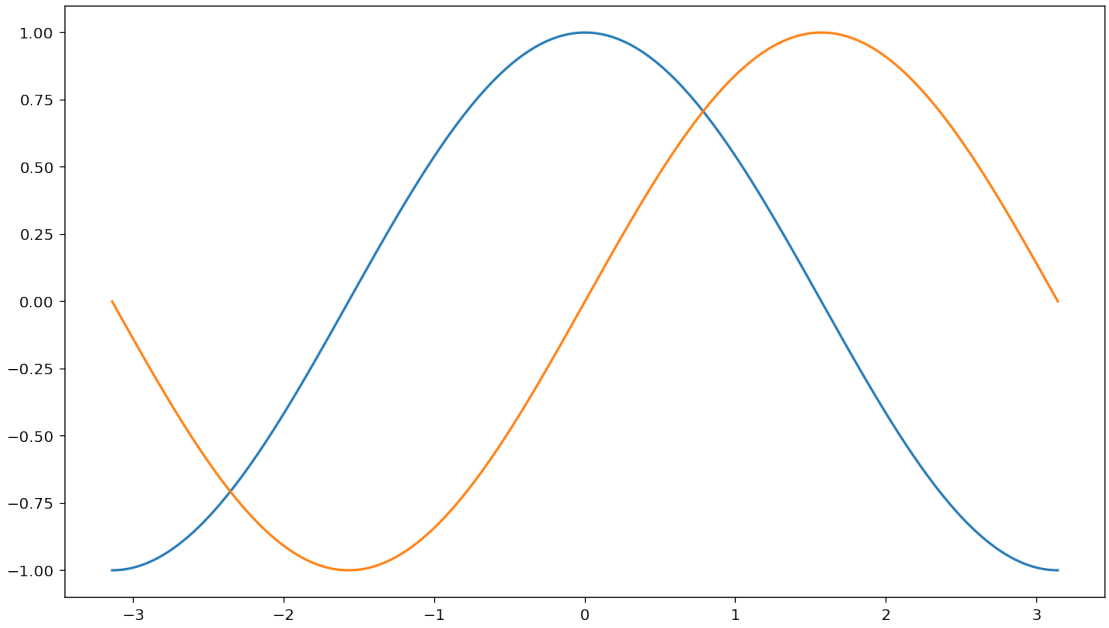
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C,S = np.cos(X), np.sin(X)

plt.plot(X,C)
plt.plot(X,S)

plt.show()
```

Out[1]:




```
In [2]: np.cos(X)

Out[2]: array([-1.        , -0.99969645, -0.99878599, -0.99726917, -0.99514692,
               -0.99242051, -0.98909161, -0.98516223, -0.98063477, -0.97551197,
               -0.96979694, -0.96349314, -0.95660442, -0.94913494, -0.94108925,
               -0.93247223, -0.92328911, -0.91354546, -0.9032472 , -0.89240058,
               -0.88101219, -0.86908895, -0.85663808, -0.84366715, -0.83018403,
               -0.81619691, -0.80171428, -0.78674494, -0.77129796, -0.75538273,
               -0.73900892, -0.72218645, -0.70492555, -0.68723669, -0.66913061,
               -0.6506183 , -0.63171101, -0.6124202 , -0.5927576 , -0.57273514,
               -0.55236497, -0.53165947, -0.51063119, -0.48929292, -0.46765759,
               -0.44573836, -0.42354851, -0.40110153, -0.37841105, -0.35549083,
               -0.3323548 , -0.30901699, -0.28549159, -0.26179286, -0.2379352 ,
               -0.21393308, -0.18980109, -0.16555388, -0.14120615, -0.1167727 ,
               -0.09226836, -0.067708  , -0.04310654, -0.0184789 ,  0.00615995,
                0.03079506,  0.05541147,  0.07999425,  0.10452846,  0.12899922,
                0.15339165,  0.17769097,  0.20188241,  0.22595129,  0.24988299,
                0.27366299,  0.29727685,  0.32071024,  0.34394892,  0.36697879,
                0.38978587,  0.41235632,  0.43467642,  0.45673264,  0.47851157,
                0.5       ,  0.52118488,  0.54205336,  0.56259275,  0.5827906 ,
                0.60263464,  0.62211282,  0.64121331,  0.65992453,  0.67823512,
                0.69613395,  0.71361015,  0.73065313,  0.74725253,  0.76339828,
                0.77908057,  0.79428989,  0.80901699,  0.82325295,  0.83698911,
                0.85021714,  0.862929  ,  0.87511698,  0.88677369,  0.89789203,
                0.90846527,  0.91848699,  0.92795109,  0.93685184,  0.94518383,
                0.952942  ,  0.96012165,  0.9667184 ,  0.97272827,  0.9781476 ,
```

```
                0.9829731 ,  0.98720184,  0.99083125,  0.99385914,  0.99628365,
                0.99810333,  0.99931706,  0.99992411,  0.99992411,  0.99931706,
                0.99810333,  0.99628365,  0.99385914,  0.99083125,  0.98720184,
                0.9829731 ,  0.9781476 ,  0.97272827,  0.9667184 ,  0.96012165,
                0.952942  ,  0.94518383,  0.93685184,  0.92795109,  0.91848699,
                0.90846527,  0.89789203,  0.88677369,  0.87511698,  0.862929  ,
                0.85021714,  0.83698911,  0.82325295,  0.80901699,  0.79428989,
                0.77908057,  0.76339828,  0.74725253,  0.73065313,  0.71361015,
                0.69613395,  0.67823512,  0.65992453,  0.64121331,  0.62211282,
                0.60263464,  0.5827906 ,  0.56259275,  0.54205336,  0.52118488,
                0.5       ,  0.47851157,  0.45673264,  0.43467642,  0.41235632,
                0.38978587,  0.36697879,  0.34394892,  0.32071024,  0.29727685,
                0.27366299,  0.24988299,  0.22595129,  0.20188241,  0.17769097,
                0.15339165,  0.12899922,  0.10452846,  0.07999425,  0.05541147,
                0.03079506,  0.00615995, -0.0184789 , -0.04310654, -0.067708  ,
               -0.09226836, -0.1167727 , -0.14120615, -0.16555388, -0.18980109,
               -0.21393308, -0.2379352 , -0.26179286, -0.28549159, -0.30901699,
               -0.3323548 , -0.35549083, -0.37841105, -0.40110153, -0.42354851,
               -0.44573836, -0.46765759, -0.48929292, -0.51063119, -0.53165947,
               -0.55236497, -0.57273514, -0.5927576 , -0.6124202 , -0.63171101,
               -0.6506183 , -0.66913061, -0.68723669, -0.70492555, -0.72218645,
               -0.73900892, -0.75538273, -0.77129796, -0.78674494, -0.80171428,
               -0.81619691, -0.83018403, -0.84366715, -0.85663808, -0.86908895,
               -0.88101219, -0.89240058, -0.9032472 , -0.91354546, -0.92328911,
               -0.93247223, -0.94108925, -0.94913494, -0.95660442, -0.96349314,
               -0.96979694, -0.97551197, -0.98063477, -0.98516223, -0.98909161,
               -0.99242051, -0.99514692, -0.99726917, -0.99878599, -0.99969645, -1.        ])
```

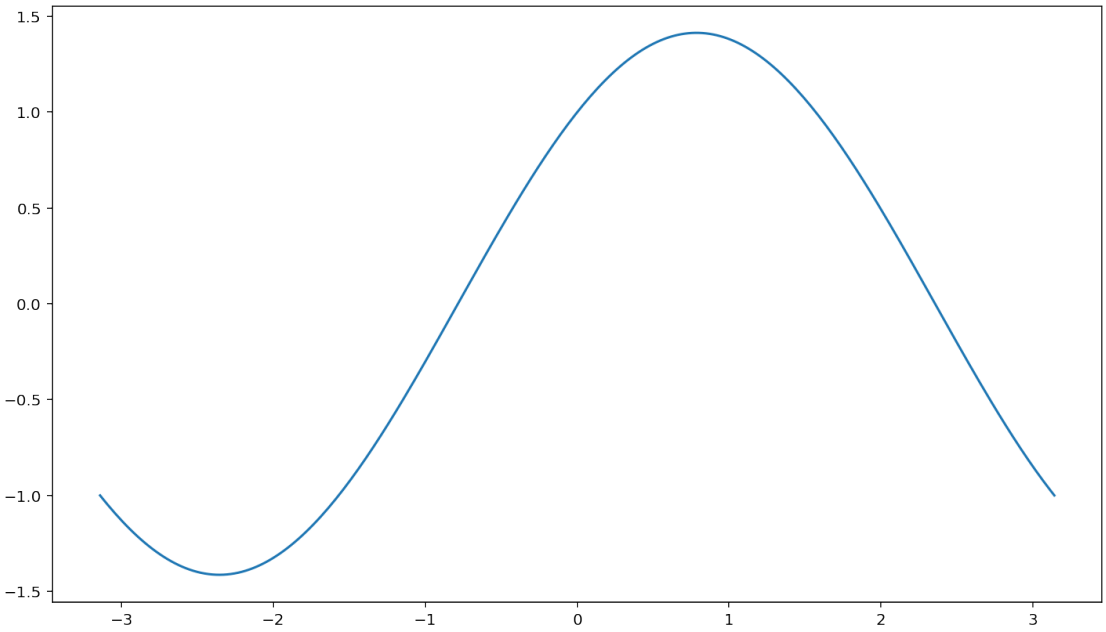
In [3]: X

```
Out[3]: array([-3.14159265, -3.11695271, -3.09231277, -3.06767283, -3.04303288,
               -3.01839294, -2.993753  , -2.96911306, -2.94447311, -2.91983317,
               -2.89519323, -2.87055329, -2.84591335, -2.8212734 , -2.79663346,
               -2.77199352, -2.74735358, -2.72271363, -2.69807369, -2.67343375,
               -2.64879381, -2.62415386, -2.59951392, -2.57487398, -2.55023404,
               -2.52559409, -2.50095415, -2.47631421, -2.45167427, -2.42703432,
               -2.40239438, -2.37775444, -2.3531145 , -2.32847456, -2.30383461,
               -2.27919467, -2.25455473, -2.22991479, -2.20527484, -2.1806349 ,
               -2.15599496, -2.13135502, -2.10671507, -2.08207513, -2.05743519,
               -2.03279525, -2.0081553 , -1.98351536, -1.95887542, -1.93423548,
               -1.90959553, -1.88495559, -1.86031565, -1.83567571, -1.81103577,
               -1.78639582, -1.76175588, -1.73711594, -1.712476  , -1.68783605,
               -1.66319611, -1.63855617, -1.61391623, -1.58927628, -1.56463634,
               -1.5399964 , -1.51535646, -1.49071651, -1.46607657, -1.44143663,
               -1.41679669, -1.39215674, -1.3675168 , -1.34287686, -1.31823692,
               -1.29359698, -1.26895703, -1.24431709, -1.21967715, -1.19503721,
               -1.17039726, -1.14575732, -1.12111738, -1.09647744, -1.07183749,
               -1.04719755, -1.02255761, -0.99791767, -0.97327772, -0.94863778,
```

```
       -0.92399784, -0.8993579 , -0.87471795, -0.85007801, -0.82543807,
       -0.80079813, -0.77615819, -0.75151824, -0.7268783 , -0.70223836,
       -0.67759842, -0.65295847, -0.62831853, -0.60367859, -0.57903865,
       -0.5543987 , -0.52975876, -0.50511882, -0.48047888, -0.45583893,
       -0.43119899, -0.40655905, -0.38191911, -0.35727916, -0.33263922,
       -0.30799928, -0.28335934, -0.2587194 , -0.23407945, -0.20943951,
       -0.18479957, -0.16015963, -0.13551968, -0.11087974, -0.0862398 ,
       -0.06159986, -0.03695991, -0.01231997,  0.01231997,  0.03695991,
        0.06159986,  0.0862398 ,  0.11087974,  0.13551968,  0.16015963,
        0.18479957,  0.20943951,  0.23407945,  0.2587194 ,  0.28335934,
        0.30799928,  0.33263922,  0.35727916,  0.38191911,  0.40655905,
        0.43119899,  0.45583893,  0.48047888,  0.50511882,  0.52975876,
        0.5543987 ,  0.57903865,  0.60367859,  0.62831853,  0.65295847,
        0.67759842,  0.70223836,  0.7268783 ,  0.75151824,  0.77615819,
        0.80079813,  0.82543807,  0.85007801,  0.87471795,  0.8993579 ,
        0.92399784,  0.94863778,  0.97327772,  0.99791767,  1.02255761,
        1.04719755,  1.07183749,  1.09647744,  1.12111738,  1.14575732,
        1.17039726,  1.19503721,  1.21967715,  1.24431709,  1.26895703,
        1.29359698,  1.31823692,  1.34287686,  1.3675168 ,  1.39215674,
        1.41679669,  1.44143663,  1.46607657,  1.49071651,  1.51535646,
        1.5399964 ,  1.56463634,  1.58927628,  1.61391623,  1.63855617,
        1.66319611,  1.68783605,  1.712476  ,  1.73711594,  1.76175588,
        1.78639582,  1.81103577,  1.83567571,  1.86031565,  1.88495559,
        1.90959553,  1.93423548,  1.95887542,  1.98351536,  2.0081553 ,
        2.03279525,  2.05743519,  2.08207513,  2.10671507,  2.13135502,
        2.15599496,  2.1806349 ,  2.20527484,  2.22991479,  2.25455473,
        2.27919467,  2.30383461,  2.32847456,  2.3531145 ,  2.37775444,
        2.40239438,  2.42703432,  2.45167427,  2.47631421,  2.50095415,
        2.52559409,  2.55023404,  2.57487398,  2.59951392,  2.62415386,
        2.64879381,  2.67343375,  2.69807369,  2.72271363,  2.74735358,
        2.77199352,  2.79663346,  2.8212734 ,  2.84591335,  2.87055329,
        2.89519323,  2.91983317,  2.94447311,  2.96911306,  2.993753  ,
        3.01839294,  3.04303288,  3.06767283,  3.09231277,  3.11695271,
        3.14159265])
```

In [4]:
```
test=np.cos(X)+np.sin(X)
plt.plot(X,test)
plt.show()
```

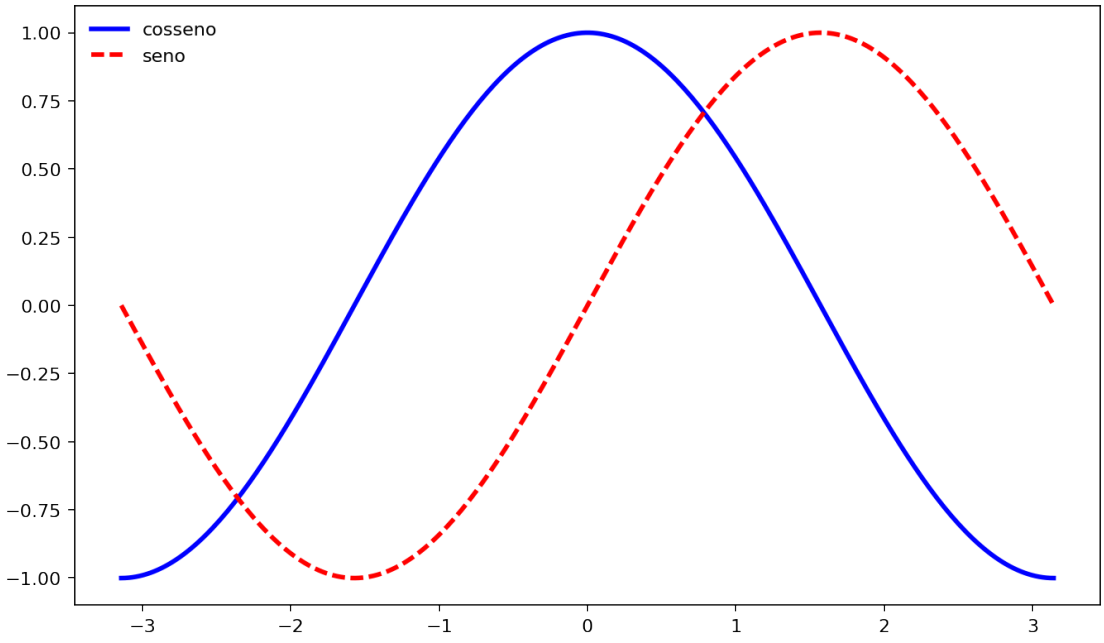
Out[4]:




**Alterando as cores e as larguras de linha, legenda:** - Primeiro passo, queremos ter o cosseno em azul eo seno em vermelho e uma linha ligeiramente mais grossa para ambos. Também alteraremos ligeiramente o tamanho da figura para torná-lo mais horizontal. - Vamos adicionar uma legenda no canto superior esquerdo. Isso só requer a adição do rótulo de argumento de palavra-chave (que será usado na caixa de legenda) para os comandos de plotagem.

```
In [5]: plt.figure(figsize=(10,6), dpi=80)
        plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosseno")
        plt.plot(X, S, color="red",  linewidth=2.5, linestyle="--", label="seno")
        plt.legend(loc='upper left', frameon=False)
        plt.show()
```

```
Out[5]:
```



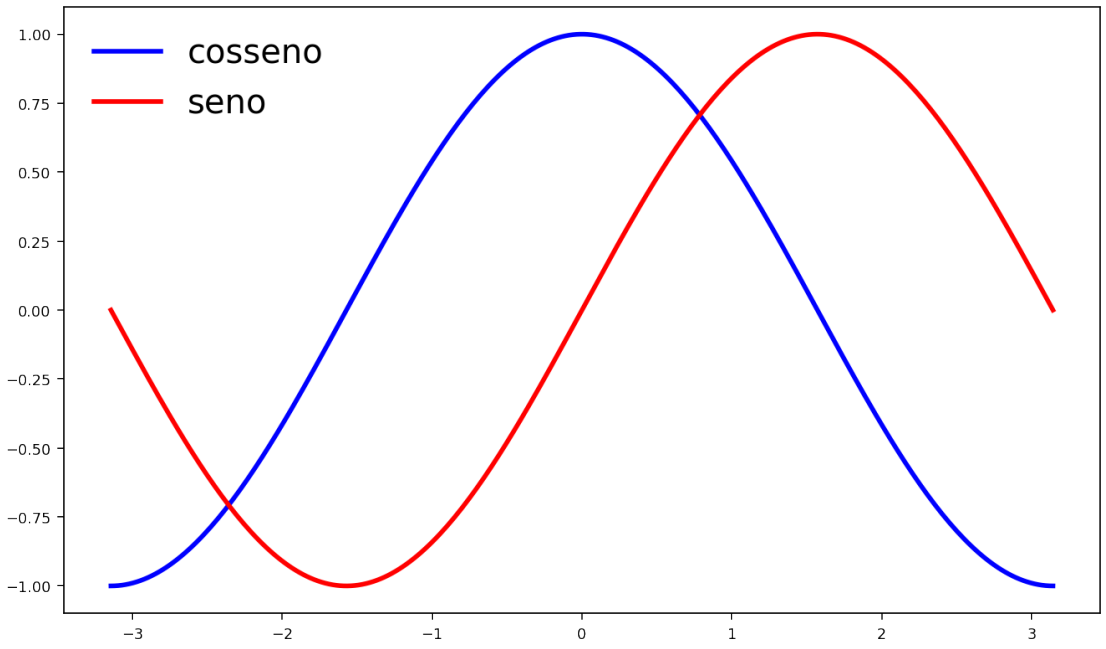

Para ajustar os tamanhos das fontes

```
In [6]: SMALL_SIZE = 8
        MEDIUM_SIZE = 10
        BIGGER_SIZE = 18
        plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
        plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
        plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
        plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
        plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
        plt.rc('legend', fontsize=BIGGER_SIZE)    # legend fontsize
        plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title

In [7]: plt.figure(figsize=(10,6), dpi=80)
        plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosseno")
        plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
        plt.legend(loc='upper left', frameon=False)

        plt.show()

Out[7]:
```




**2. Carregar e examinar o conjunto de dados** - Vamos começar. A função mostrada na célula abaixo carrega os dados do arquivo .csv. Uma limpeza mínima é realizada nos dados. As linhas com valores em falta são eliminadas e algumas colunas são convertidas a partir de cadeias contendo números para dados numéricos utilizando-se do Pandas.

```
In [8]: def read_auto_data(fileName = "Automobile price data.csv"):
            'Function to load the auto price data set from a .csv file'
            import pandas as pd
            import numpy as np

            ## Read the .csv file with the pandas read_csv method
            auto_prices = pd.read_csv(fileName)

            ## Remove rows with missing values, accounting for mising values coded as '?'
            cols = ['price', 'bore', 'stroke',
                  'horsepower', 'peak-rpm']
            for column in cols:
                auto_prices.loc[auto_prices[column] == '?', column] = np.nan
            auto_prices.dropna(axis = 0, inplace = True)

            ## Convert some columns to numeric values
            for column in cols:
                auto_prices[column] = pd.to_numeric(auto_prices[column])
        #    auto_prices[cols] = auto_prices[cols].as_type(int64)

            return auto_prices
        auto_prices = read_auto_data()
```

```
In [9]: auto_prices.head()
```

- Você pode ver que há tipos de variáveis numéricas e de string (categóricas).
- Como uma próxima etapa examine algumas estatísticas de resumo das colunas numéricas usando o método Pandas descrever.
- Note que as unidades de medida são americanas e não métricas.

```
In [10]: auto_prices.describe()
```

**2.1- Tipos de gráficos básicos** - Agora que nós carregamos e tivemos um primeiro olhar para os dados, vamos começar a trabalhar fazendo alguns gráficos. - Existem tipos de gráfico enumeráveis que são usados para a exploração de dados.

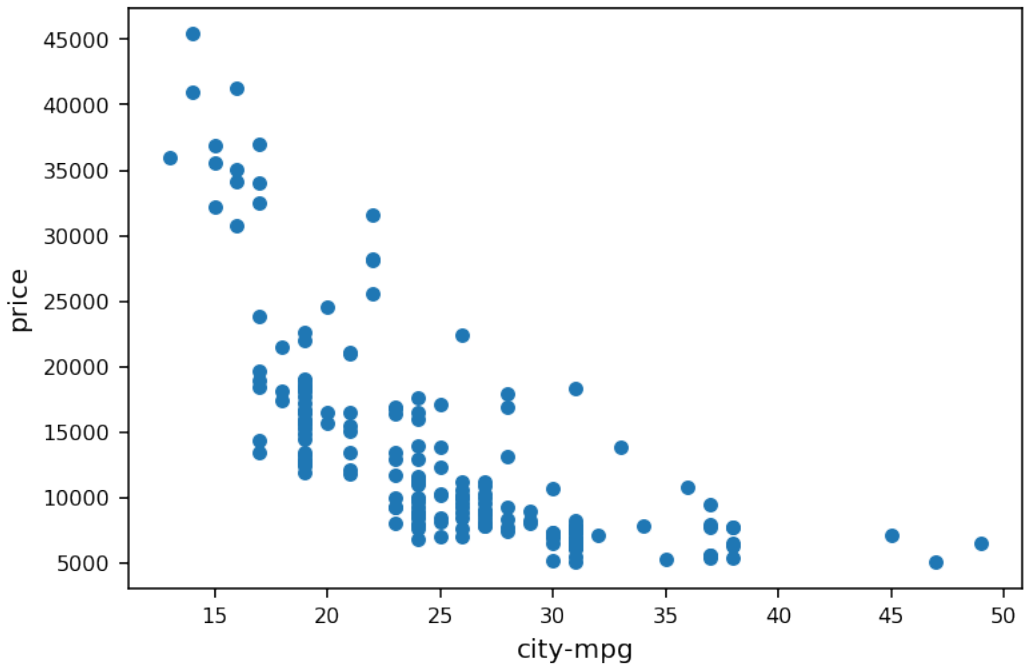
Nesta disciplina focaremos nos seguintes tipos de gráficos: a) Gráfico de dispersão b) Gráficos em Linha c) Gráficos em Barras d) Histogramas e) BoxPlot f) BoxPlot usando Densidade do Kernel g) ViolinPlot

** a) Gráficos de dispersão** - Os gráficos de dispersão mostram a relação entre duas variáveis sob a forma de pontos no gráfico. Em termos simples, os valores ao longo de um eixo horizontal são plotados contra um eixo vertical. - O pacote Pandas contém um número de métodos de plotagem úteis que operam em dataframes. A receita simples para traçar a partir de um dataframe Pandas - Use o método de plotagem, especificando o argumento de tipo ou use um método de plotagem gráfico-específico. - Especifique as colunas com os valores para os eixos x e y.

```
In [11]: %matplotlib inline
         auto_prices.plot(kind = 'scatter', x = 'city-mpg', y = 'price')
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6870e0ff60>
```

```
Out[11]:
```



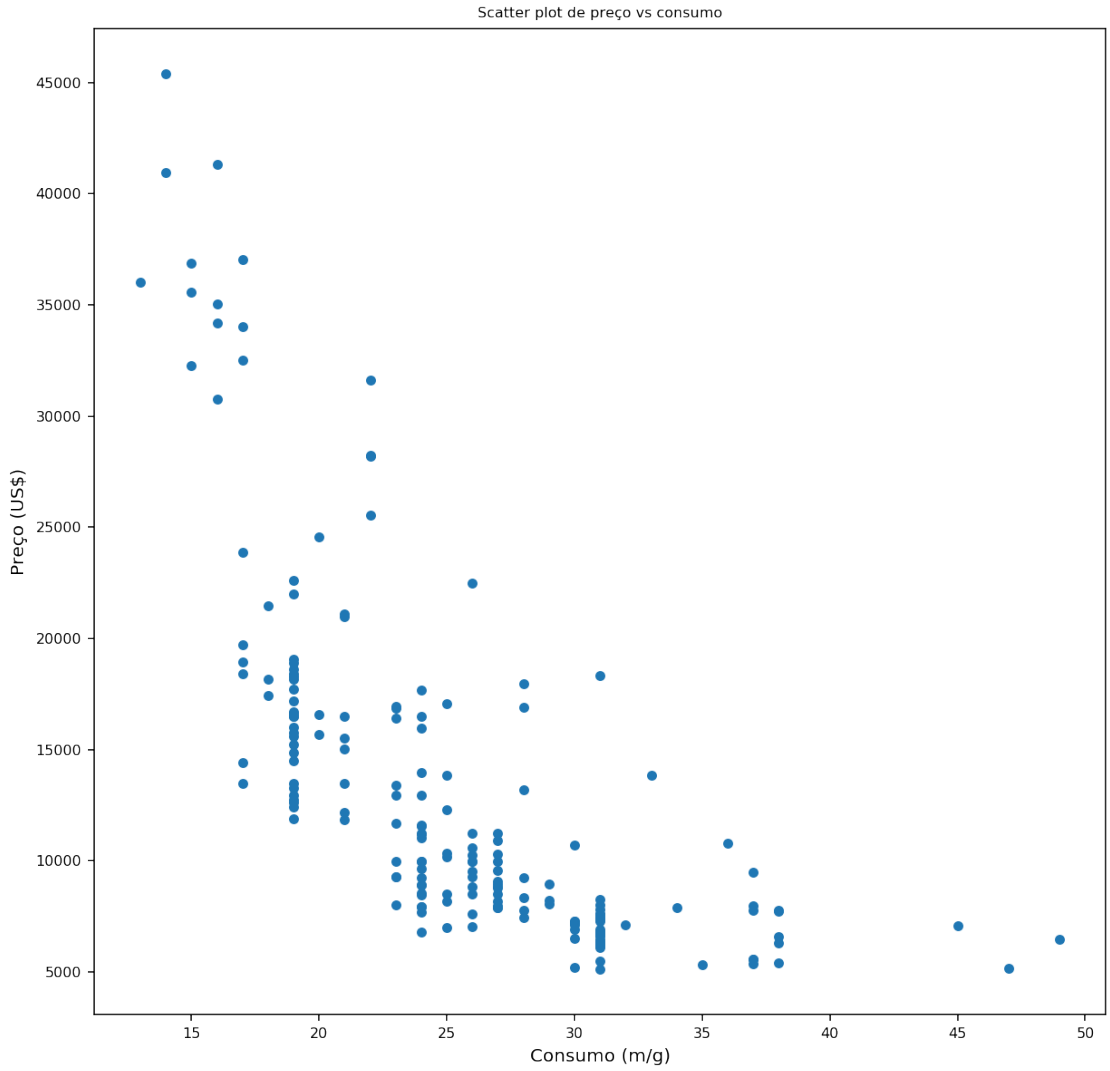
- Nossa receita básica funcionou, mas não nos dá muito controle sobre o resultado.
- Aproveitamos o fato de que os métodos Pandas plot são construídos em matplotlib. Assim, podemos especificar uma figura e um ou mais eixos dentro dessa figura. Podemos controlar muitos atributos do plot uma vez especificados os eixos. Nossa receita melhorada se parece com isto:

**- Especifique uma figura, incluindo um tamanho.**
**- Defina um ou mais eixos dentro da figura.**
**- Use o método de escolha de panda. Certifique-se de especificar o eixo que você está usando.**
**Use métodos nos eixos para controlar atributos como títulos e rótulos de eixo.**

```
In [12]: import matplotlib.pyplot as plt
         fig = plt.figure(figsize=(10, 10)) # define plot area
         ax = fig.gca() # define axis
         auto_prices.plot(kind = 'scatter', x = 'city-mpg', y = 'price', ax = ax)
         ax.set_title('Scatter plot de preço vs consumo') # Give the plot a main title
         ax.set_xlabel('Consumo (m/g)') # Set text for the x axis
         ax.set_ylabel('Preço (US$)')# Set text for y axis

Out[12]: <matplotlib.text.Text at 0x7f685f581eb8>

Out[12]:
```




- Na trama acima, podemos ver que os carros mais caros têm a menor eficiência de combustível, enquanto carros relativamente baratos também são mais econômicos para a unidade.

**b) Gráfico em Linha** - Os gráficos de linhas são semelhantes aos gráficos de pontos. Em traçados de linha os pontos discretos são conectados por linhas. - Primeiro, vamos criar um dataframe, com uma relação simples entre x e y.

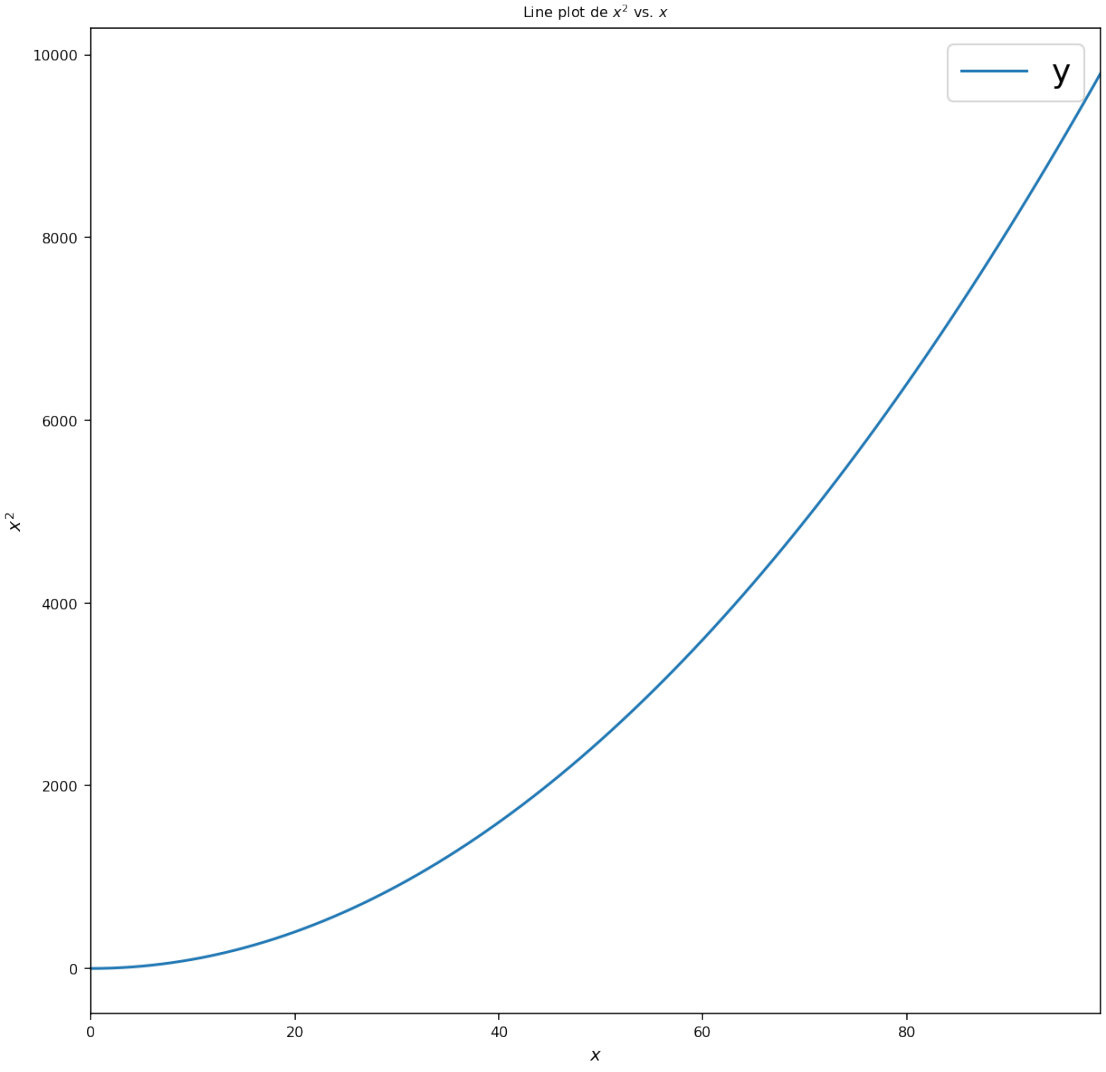
```
In [13]: import pandas as pd
         x = list(range(100))
         y = [z * z for z in range(100)]
         df = pd.DataFrame({'x':x, 'y':y})
```

- Agora aplique a receita anterior para plotar x vs y e gerar um gráfico em linha.

```
In [14]: fig = plt.figure(figsize=(10, 10)) # define área do plot
         ax = fig.gca() # define eixo
         df.plot(x = 'x', y = 'y', ax = ax) ## linha é o formato padrão
         ax.set_title('Line plot de $x^2$ vs. $x$') # Título Principal
         ax.set_xlabel('$x$') # Eixo x
         ax.set_ylabel('$x^2$')# Eixo y

Out[14]: <matplotlib.text.Text at 0x7f685bcd3d68>

Out[14]:
```




**c) Gráfico de Barras** - Gráficos de barras são usados para exibir as contagens de valores exclusivos de uma variável categórica. A altura da barra representa a contagem para cada categoria única da variável. - É improvável que o quadro de dados do pandas inclua contagens por categoria de uma variável. Assim, o primeiro passo para fazer um gráfico de barras é calcular as

contagens. Felizmente, pandas tem um método ***value_counts***. O código abaixo usa esse método para criar um novo quadro de dados contendo as contagens por marca do carro.

```
In [15]: counts = auto_prices['make'].value_counts() # encontre a contagem para cada categoria
         counts
```

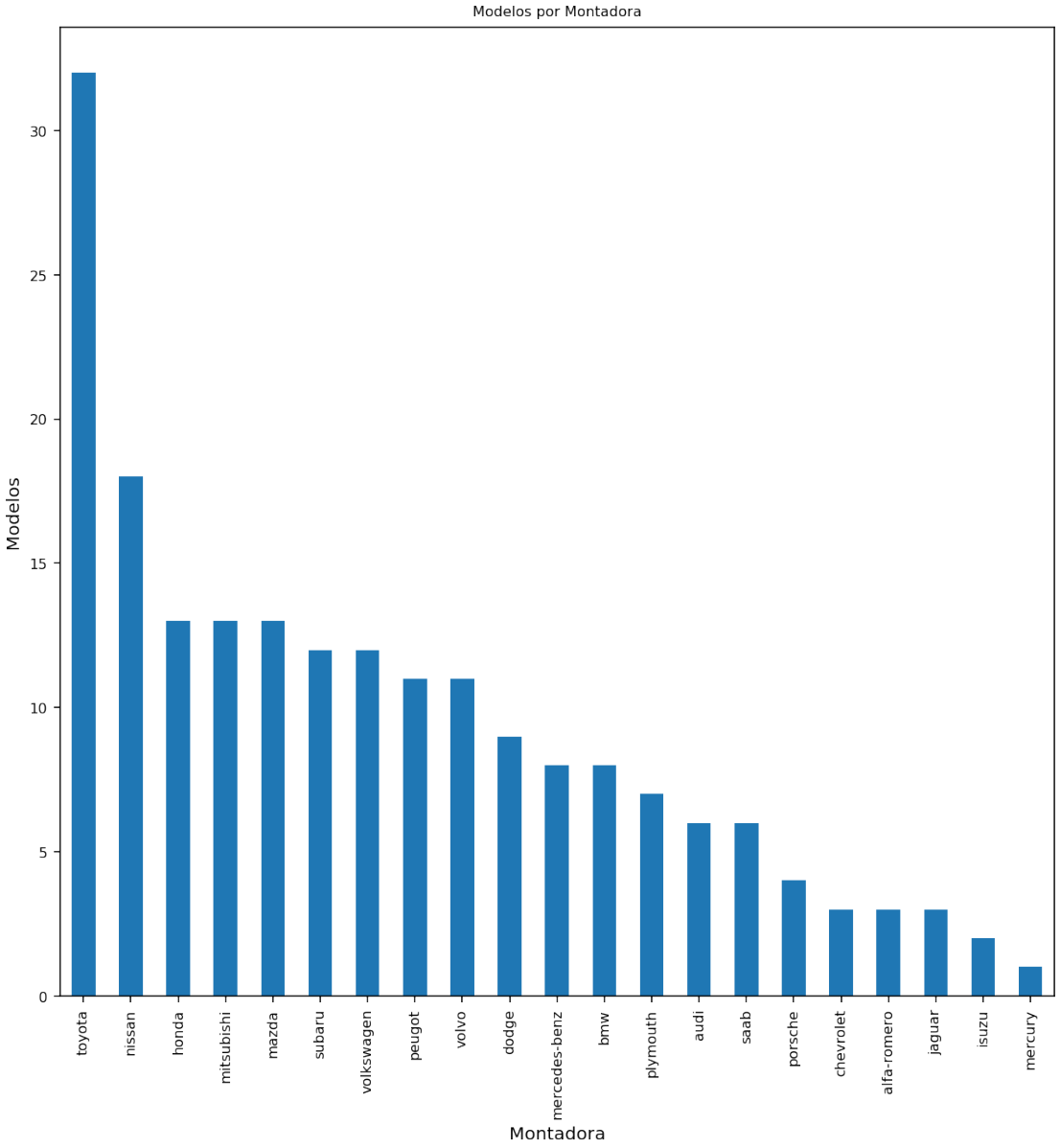
```
Out[15]: toyota           32
         nissan           18
         honda            13
         mitsubishi       13
         mazda            13
         subaru           12
         volkswagen       12
         peugot           11
         volvo            11
         dodge             9
         mercedes-benz     8
         bmw               8
         plymouth          7
         audi              6
         saab              6
         porsche           4
         chevrolet         3
         alfa-romero       3
         jaguar            3
         isuzu             2
         mercury           1
         Name: make, dtype: int64
```

- Você pode ver a lista ordenada de contagens descrita acima. Observe que esse quadro de dados é muito menor do que o original, uma vez que só precisa de uma linha para cada valor exclusivo da marca.
- Agora, faça o gráfico de barras usando o quadro de dados de contagens, crie o gráfico de barras do número de carros por marca. Observe que a receita para este gráfico é a mesma que usamos para gráficos de dispersão e gráficos de linha, usando apenas o método .plot.bar.

```
In [16]: fig = plt.figure(figsize=(10,10)) # define plot area
         ax = fig.gca() # define axis
         counts.plot.bar(ax = ax) # Use the plot.bar method on the counts data frame
         ax.set_title('Modelos por Montadora') # Give the plot a main title
         ax.set_xlabel('Montadora') # Set text for the x axis
         ax.set_ylabel('Modelos')# Set text for y axis
```

```
Out[16]: <matplotlib.text.Text at 0x7f685bccbcf8>
```

```
Out[16]:
```




- O gráfico de barra mostra claramente quais fabricantes de automóveis têm o maior número de modelos. As marcas mais especializadas têm relativamente menos modelos.

### 1.0.1 Exercício1

- A "classificação" de uma palavra é a sua posição em uma lista de palavras classificadas por frequência: a palavra mais comum tem a classificação 1, a segunda mais comum é 2 etc.
- A lei de Zipf descreve a relação entre classificações e frequências das palavras em linguagens naturais (http://en.wikipedia.org/wiki/Zipf's_law). Ela prevê especificamente que a frequência, $f$, da palavra com classificação $ r $ é:

$$f = crs$$

- onde s e c são parâmetros que dependem do idioma e do texto. Se você tomar o logaritmo de ambos os lados desta equação, obtemos:

$$\log f = \log cs \log r$$

- Se você traçar o log de f contra o log de r, terá uma linha reta com uma elevação -s e interceptar o log de c.

**- Escreva um programa que leia um texto em um arquivo, conte as frequências das palavras e exiba uma linha para cada palavra, em ordem descendente da frequência, com log de f e log de r. Use o programa gráfico de sua escolha para traçar os resultados e verifique se formam uma linha reta. Você pode estimar o valor de s?**

### 1.0.2 Exercício 2

Agora que você já viu como criar alguns plots simples, é sua vez de realizar uma visualização. Crie o seguinte gráfico de dispersão:

- Traçar o tamanho do motor contra o preço.
- Defina o tamanho da figura como 8 x 8.
- Forneça um título significativo, rótulo do eixo x e rótulo do eixo y.

### 1.0.3 Exercício 3

- Faça um gráfico de Barras com os dados contidos no DataFrame do exercício dos epsódios do Pokemon. Represente o número de episódios para cada Temporada. Neste exercicio o gráfico deve conter legenda, titulo, nome dos eixos e cada barra deve conter uma cor diferente.