Felipe Galvão

Web Development / Desenvolvimento Web

Visualização de Dados com Python (matplotlib)

Felipe Galvão – Python (Português) – março 8, 2016

English version of this post / Versão em inglês deste post

Introdução

A Visualização de Dados é uma vertente essencial da Análise de Dados. Visualizações inteligentes permitem novas descobertas, insights e conhecimento através de gráficos e tabelas selecionados.

Existem muitas ferramentas para Visualização de Dados, e a cada dia novas ferramentas surgem. Nós iremos focar na ferramenta que é provavelmente a mais difundida para este fim no Python, que é o matplotlib.

O matplotlib

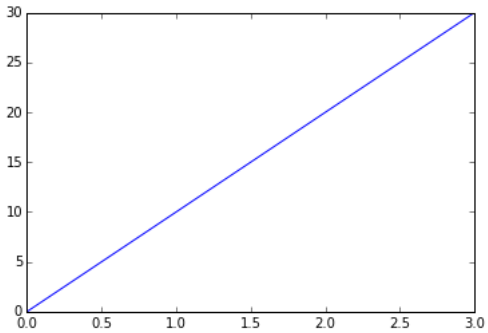
O matplotlib (http://matplotlib.org/) é uma biblioteca do Python para criação de gráficos em 2D, bastante utilizada para visualização de dados e que apresenta uma série de possibilidades gráficas, como gráficos de barra, linha, pizza, histogramas, entre muitos outros.

O matplotlib já está incluso na distribuição Anaconda (e em praticamente todas as distribuições de Python focadas em Análise de Dados e programação científica). Então, se você a está utilizando, não há necessidade de qualquer instalação. Mas, caso precise instala-lo, segue o link com instruções para instalação do Matplotlib.

Traçando os primeiros gráficos

Bem, vamos começar com um exemplo bem simples. Em primeiro lugar, é uma convenção usualmente adotada importar a coleção de comandos pyplot do matplotlib. De acordo com a documentação oficial, esta coleção de comandos faz com que o matplotlib funcione em estilo semelhante ao MATLAB. Normalmente, importa-se esta coleção como plt. Feito isso, no matplotlib, cada comando executa uma alteração no gráfico, como criação da área, traçado dos pontos, mudança do label nos eixos, e o comando final exibe o gráfico. Vamos passar uma lista para o comando plot() e depois usar o comando show() para exibir o gráfico.

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

plt.plot([0,10,20,30])
plt.show()
```

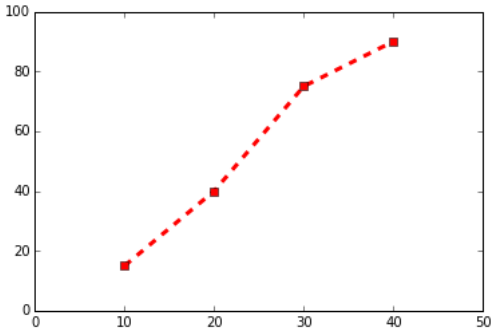


matplotlib – gráfico de linha 1

Por padrão, ao receber uma lista, o comando plot traça um gráfico de linha, com o índice da lista no eixo X e os itens no eixo Y, como podemos ver acima.

Para o segundo exemplo, vamos definir os pontos do eixo X. Adicionalmente, um terceiro argumento passado à função plot() define o formato dos pontos e da linha do gráfico. Vamos fazer o gráfico com quadrados vermelhos e uma linha tracejada, passando como argumentos color, linestyle e marker. Usaremos "r" para a cor red, linestyle "–" para o tracejado e marker "s" para square, ou seja, quadrado. Por fim, vamos deixar a linha mais grossa com o parâmetro linewidth. Vamos também definir o eixo com a função axis, para melhorar um pouco a visualização. No axis você passa, dentro de uma lista, os valores para o eixo X e depois para o eixo Y, indicando os pontos iniciais e finais para cada eixo:

```
plt.plot([10,20,30,40], [15, 40, 75, 90], linestyle='--', color='r', marker='s',
         linewidth=3.0)
plt.axis([0,50,0,100])
plt.show()
```

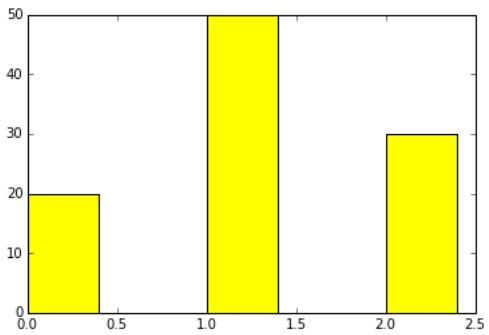


matplotlib – gráfico de linha 2

Gráfico de Barras

Outro gráfico popular presente no matplotlib é o gráfico de barras. Para o gráfico de barras, usamos a função bar(), onde definimos a posição das barras no eixo X e sua altura no eixo Y. Adicionalmente, também podemos configurar outras características, como a espessura das barras, cor, entre outros. O eixo X será um range com a mesma quantidade de itens do eixo Y. Vejamos um exemplo bem simples, onde vamos guardar as configurações em variáveis e então passa-los para a função.

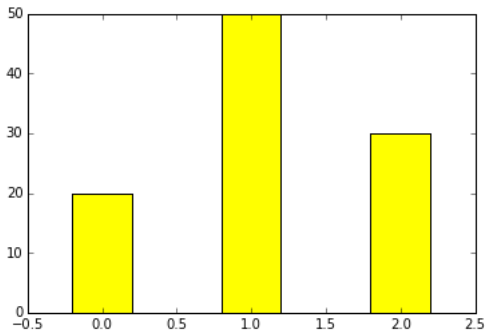
```
# Variáveis para o Bar Chart
y_axis = [20,50,30]
x_axis = range(len(y_axis))
width_n = 0.4
bar_color = 'yellow'

plt.bar(x_axis, y_axis, width=width_n, color=bar_color)
plt.show()
```



matplotlib – gráfico de barras 1

Com o parâmetro align='center', as barras ficarão centralizadas nas posições definidas para o eixo X, conforme o exemplo abaixo:

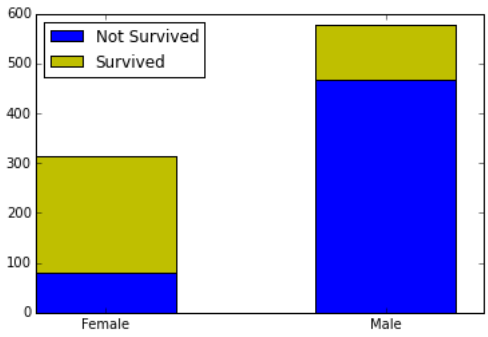
```
plt.bar(x_axis, y_axis, width=width_n, color=bar_color, align='center')
plt.show()
```



matplotlib – gráfico de barras centralizado

Agora vamos usar o Dataset do Titanic (veja aqui) para fazer um gráfico de barras empilhadas, onde poderemos visualizar para cada sexo os sobreviventes. Primeiro usaremos a função pivot_table() do Pandas para criar uma tabela que agregue estes dados mencionados, passando qual deve ser a variável das linhas e colunas, a função que deve agregar os valores (pode ser soma, contagem, entre outros) e os valores aos quais esta função será aplicada, dividindo entre os valores já definidos nas linhas e colunas. Para quem já trabalhou com tabela dinâmica no Excel, esta função fica mais fácil de entender. Depois usaremos os dados para fazer o gráfico de barras empilhadas:

```
tabela = pd.pivot_table(data=train_df, values='PassengerId', index='Sex', columns='Survived', aggfunc='c
print(tabela)
Survived    0    1
Sex
female     81  233
male      468  109

# Array com os não-sobreviventes, divididos em male e female
bar_1 = tabela[0]
# Array com os sobreviventes, divididos em male e female
bar_2 = tabela[1]
# Range com a quantidade de itens das barras
x_pos = np.arange(len(bar_1))

first_bar = plt.bar(x_pos, bar_1, 0.5, color='b')
second_bar = plt.bar(x_pos, bar_2, 0.5, color='y', bottom=bar_1)
# Definir posição e labels no eixo X
plt.xticks(x_pos+0.25, ('Female','Male'))

plt.show()
```



matplotlib – Gráfico de Barras empilhado

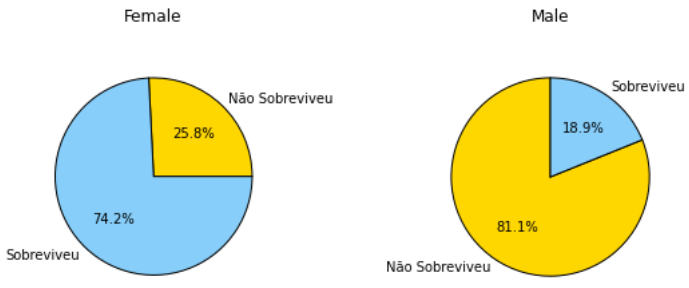
As barras são criadas com a função bar(), onde definimos as posições do começo da barra no eixo X, a altura dos itens no eixo Y, a espessura da barra e sua cor. A função xticks define labels no eixo X. Primeiro definimos suas posições e depois um Tuple com os valores a serem incluídos no gráfico.

Porém, melhor ainda para fazer esta comparação seriam dois gráficos de pizza (pie charts), mostrando percentualmente quem sobreviveu e quem não para cada sexo. Vamos aproveitar para aprender gráficos de pizza e também como fazer dois gráficos em uma mesma figura.

Gráficos de pizza e gráficos múltiplos

Para gráficos múltiplos usamos a função subplots. Ela cria o que o matplotlib chama de "figure", que seria o espaço onde os múltiplos gráficos podem ser criados. Os espaços onde são criados os gráficos funcionam como listas, ou arrays. Nos casos de uma linha ou coluna, os espaços são arrays de uma dimensão, e nos casos de mais de uma linha e coluna, os arrays são de duas dimensões. Aqui criaremos uma figure com uma linha e duas colunas, onde colocaremos nossos gráficos de pizza lado a lado usando a função pie, e definindo nela os valores que irão compor o gráfico, labels para esses valores, vamos incluir a porcentagem de cada valor relativo ao total da pizza, junto com sua formatação e as cores de cada pedaço. Depois disso, incluiremos um título em cada subplot e a opção axis('equal') garantirá que os gráficos serão redondos (meramente estético) .

```
# Cria os arrays com os valores de sobrevivência e não sobrevivência por sexo
pie_female = tabela.loc['female']
pie_male = tabela.loc['male']

# Criação da figure com uma linha e duas colunas. Figsize define o tamanho da
# figure
fig, eixos = plt.subplots(nrows=1, ncols=2, figsize=(8,4))
# Cria o gráfico de pizza na primeira posição com as configurações definidas
pie_1 = eixos[0].pie(pie_female, labels=['Não Sobreviveu','Sobreviveu'],
                     autopct='%1.1f%%', colors=['gold', 'lightskyblue'])
# Define o título deste gráfico
eixos[0].set_title('Female')
# Deixa os dois eixos iguais, fazendo com que o gráfico mantenha-se redondo
eixos[0].axis('equal')
# Idem a acima, para o segundo gráfico de pizza
pie_2 = eixos[1].pie(pie_male, labels=['Não Sobreviveu','Sobreviveu'],
                     autopct='%1.1f%%', startangle=90, colors=['gold', 'lightskyblue'])
eixos[1].set_title('Male')
plt.axis('equal')
# Ajusta o espaço entre os dois gráficos
plt.subplots_adjust(wspace=1)
plt.show()
```

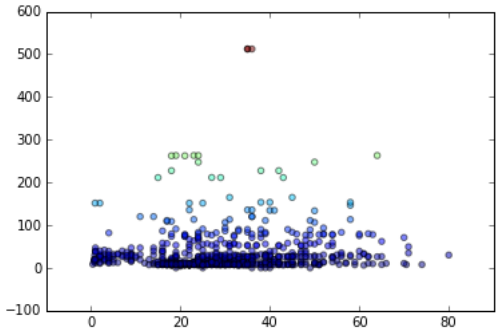


matplotlib – gráfico de pizza múltiplo

Gráfico de dispersão (ou scatter plot)

Outro gráfico que pode ser útil é o gráfico de dispersão, ou scatter plot. Ele é bastante utilizado quando queremos ver se existe alguma relação entre duas variáveis. A função do pyplot utilizada é a scatter(). Vamos verificar, por exemplo, se existe entre os passageiros do Titanic alguma relação entre a idade e o valor pago na passagem. Para a função scatter, passaremos os valores da primeira variável para o eixo X e da segunda para o eixo Y. Também definiremos um valor para "alpha", para que os pontos sejam transparentes e seja possível ver um pouco da transposição, e também vamos definir o parâmetro "c", para que as cores mudem de acordo com o valor pago no tíquete.

```
scatter_plot = plt.scatter(train_df['Age'], train_df['Fare'], alpha=0.5,
                           c=train_df['Fare'])
plt.show()
```



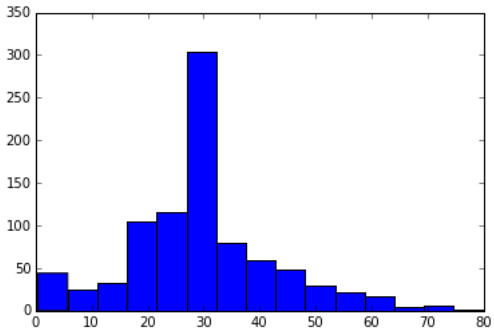
matplotlib – gráfico de dispersão

Também é possível configurar para que o tamanho dos pontos varie de acordo com o valor de uma dada variável, bastando passar a dada variável para o parâmetro 's', de size, ao invés do parâmetro 'c', de color.

Histogramas

Histogramas são úteis para visualizar a distribuição de uma série de valores. Vamos ver como se distribuem as idades dos passageiros do Titanic com um histograma. A função neste caso é a hist(). Nela, passaremos como parâmetros os valores de idade e a quantidade de barras, ou bins. A quantidade de barras é opcional, e se não passarmos nenhuma quantidade ela é definida em 10 por padrão. Também vamos fazer o preenchimento dos valores inexistentes de idade (como já explicado neste post sobre manipulação de dados) pois a função hist() retorna um erro caso haja valores inexistentes nos dados passados.

```
train_df['Age'].fillna(train_df['Age'].mean(), inplace=True)
histogram_example = plt.hist(train_df['Age'], bins=15)
plt.show()
```



matplotlib – Histograma

Existe ainda uma série de outros tipos de gráficos no matplotlib, como diagramas de caixa (boxplot), gráfico radar e outros. Mas acredito que os gráficos mostrados neste post sejam alguns dos mais úteis e utilizados na análise de dados. Em um próximo post falaremos um pouco mais sobre visualização de dados, com o Seaborn.

Fiquem ligados! 🤙

Tags ANALISE DE DADOS CIÊNCIA DE DADOS DADOS DATAFRAME PANDAS PROGRAMACAO PYTHON PYTHON BASICO VISUALIZAÇÃO DE DADOS

Manipulação de Dados com Python (Pandas) Data Visualization with Python (matplotlib)

8 comments on “Visualização de Dados com Python (matplotlib)”

Pingback: Data Visualization with Python (matplotlib) - Felipe Galvão

Pingback: Mais Visualização de Dados com Python (agora com Bokeh) - Felipe Galvão

Rafael Naves disse:

outubro 22, 2016 às 15:11

Obrigado pelo material e principalmente pela parte em português, está sendo de grande ajuda pra iniciar os estudos. Uma questão que não consegui fazer, no gráfico de pizza, como poderia colocar em cada um dos títulos o total em cada sexo?

Responder

Lucas disse:

janeiro 1, 2017 às 18:56

Muito bom, explicado de maneira simples!

Responder

Felipe Galvão disse:

janeiro 2, 2017 às 08:41

Valeu Lucas, muito obrigado!

Abçs

Responder

Luiz disse:

janeiro 17, 2017 às 13:46

Como faço para plotar pontos com 3 coordenadas, usando os eixos X, Y e Z? No Matlab uso o comando plot3. E no Python?

Responder

Pingback: DataViz - Ferramentas para visualização de dados em Python -

Pingback: DataViz – Ferramentas para visualização de dados em Python – Agency Major

Deixe uma resposta

O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

Comentário

Nome *

E-mail *

Site

PUBLICAR COMENTÁRIO

ABOUT ME / SOBRE MIM

Hi, i'm Felipe Galvão. Here, I'll talk about web development and everything I learned in my journey as a programmer.

Olá, sou Felipe Galvão. Aqui, vou falar de desenvolvimento web e tudo que aprendi na minha jornada como programador.

SOCIAL MEDIA

Pesquisar ...

PESQUISAR

TÓPICOS RECENTES

- Primeiros Passos com React
- React First Steps
- Understanding webpack and creating a React application with it
- Entendendo webpack e criando uma aplicação em React com ele

- Creating development environments and installing Django with Anaconda

COMENTÁRIOS

- React First Steps - Felipe Galvão em Primeiros Passos com React
- Primeiros Passos com React - Felipe Galvão em React First Steps
- React First Steps - Felipe Galvão em Understanding webpack and creating a React application with it
- Understanding webpack and creating a React application with it - Felipe Galvão em Entendendo webpack e criando uma aplicação em React com ele
- Entendendo webpack e criando uma aplicação em React com ele - Felipe Galvão em Understanding webpack and creating a React application with it

ARQUIVOS

- abril 2017
- março 2017
- janeiro 2017
- junho 2016
- maio 2016
- abril 2016
- março 2016
- fevereiro 2016
- dezembro 2015
- novembro 2015

CATEGORIAS

- Basic R
- Django (English)
- Django (Português)
- Javascript (English)
- Javascript (Português)
- Python (English)
- Python (Português)
- R (English)
- R (Português)
- R Básico
- React (English)
- React (Português)