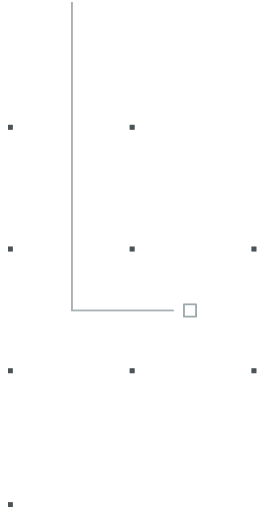


FIAP

NBA



Visualização de Dados

Dheny R. Fernandes

1. Introdução

2. Matplotlib

1. Gráfico de Barras
2. Histograma
3. Barra Empilhada
4. Scatter
5. Box-plot

3. Integrando Pandas, Matplotlib e Seaborn

Introdução



“Eu acredito que visualização é um dos meios mais poderosos de se atingir objetivos pessoais”. Harvey Mackay

Parte fundamental do kit de ferramentas de um cientista de dados é a visualização de dados. Apesar de ser **fácil criar visualizações**, é muito mais **difícil criar boas visualizações**.

Há duas principais razões para se usar visualização de dados:

- Para explorar dados
- Para comunicar dados

Nesta aula, vamos nos concentrar em construir as habilidades necessárias para começar **explorar nossos dados** e produzir visualizações que serão usadas sempre que fizermos análise de dados.

Existe uma variedade de ferramentas de visualização de dados: matplotlib, bokeh, plotly, altair, entre outros.

O foco dessa aula será no matplotlib e, depois, na integração deste com o seaborn.

Matplotlib



Matplotlib é uma biblioteca de visualização de dados projetada para criar plots de alta qualidade

Foi criada em 2002 por John Hunter para permitir uma interface em Python da biblioteca do MATLAB.

Importação:

- `Import matplotlib.pyplot as plt`

Os plots dentro do matplotlib residem dentro do objeto *Figure*. Como não é possível plotar uma imagem em branco, vamos criar alguns subplots para começar a entender a biblioteca:

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```

Agora é preciso criar gráficos a partir de alguns dados para preencher cada plot:

```
import numpy as np
from numpy.random import randn
ax1.plot(randn(50).cumsum(), 'k--')
ax2.hist(randn(100), bins=20, color='k', alpha=0.3)
ax3.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
fig
```

Entretanto, podemos criar um único plot a partir de um determinado conjunto de dados:

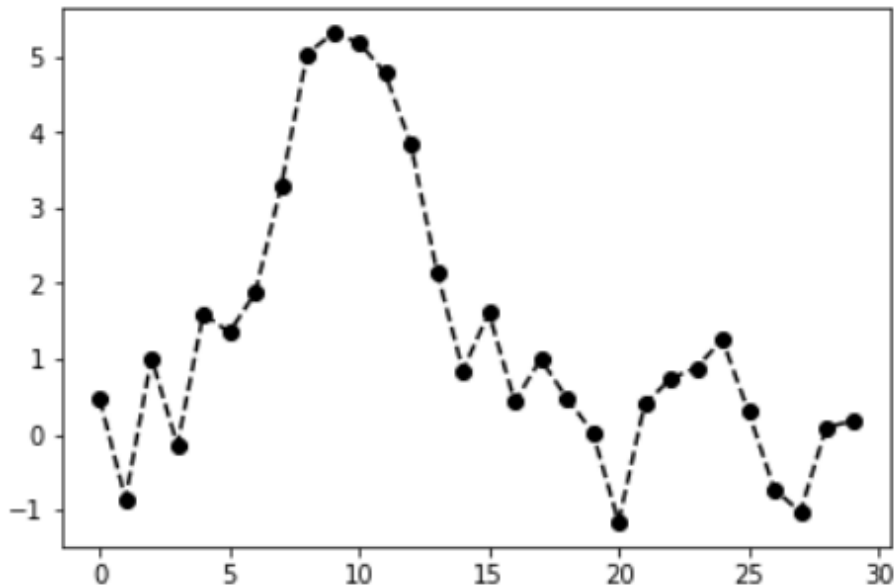
```
x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')

plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")

plt.legend()
plt.show()
```

É possível alterar cor, marcador e estilo de linha de um plot:

```
plt.plot(randn(30).cumsum(), color='k', linestyle='dashed', marker='o')
```



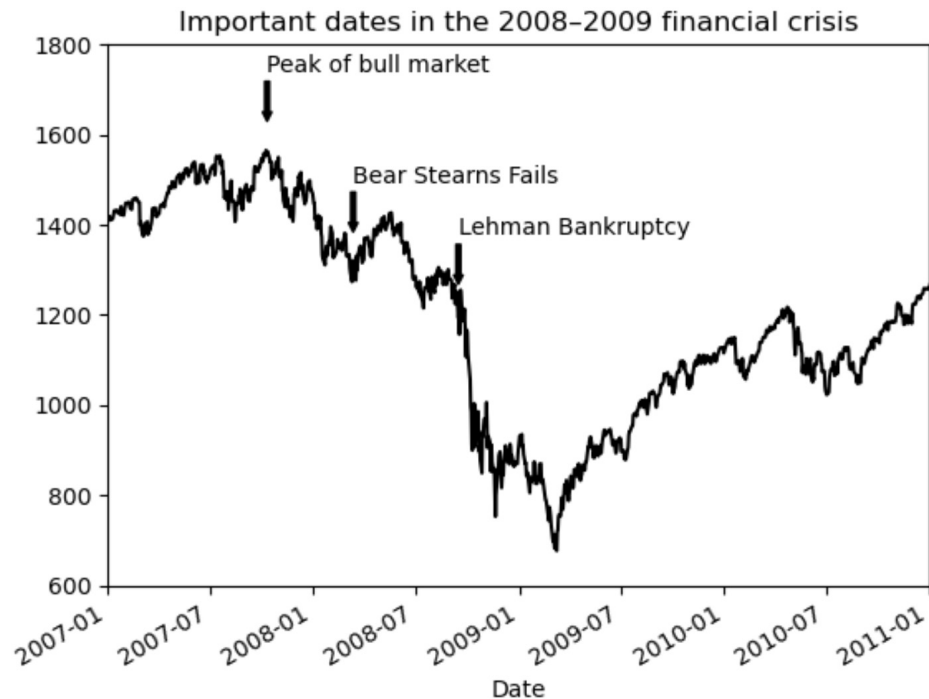
É possível ajustar o título, eixos, ticks e rótulos dos ticks:

```
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum())
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                             rotation=30, fontsize='small')
ax.set_title('My matplotlib plot')
ax.set_xlabel('Stages')
```

Legendas tem fácil manipulação também:

```
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'k', label='one')
ax.plot(randn(1000).cumsum(), 'k--', label='two')
ax.plot(randn(1000).cumsum(), 'k.', label='three')
ax.legend(loc='best')
```

Em adição aos tipos padrões de plots, é possível desenhar suas próprias anotações, que podem consistir em texto, setas e outras formas. Veja o plot abaixo e o código que o constrói no jupyter:

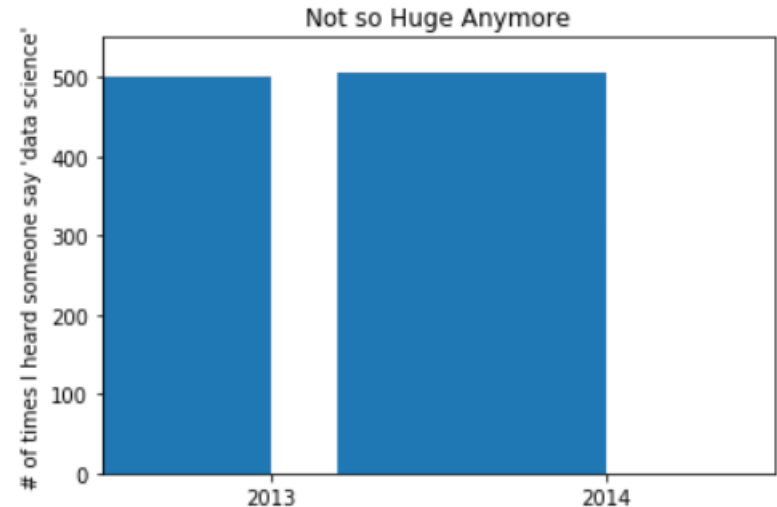
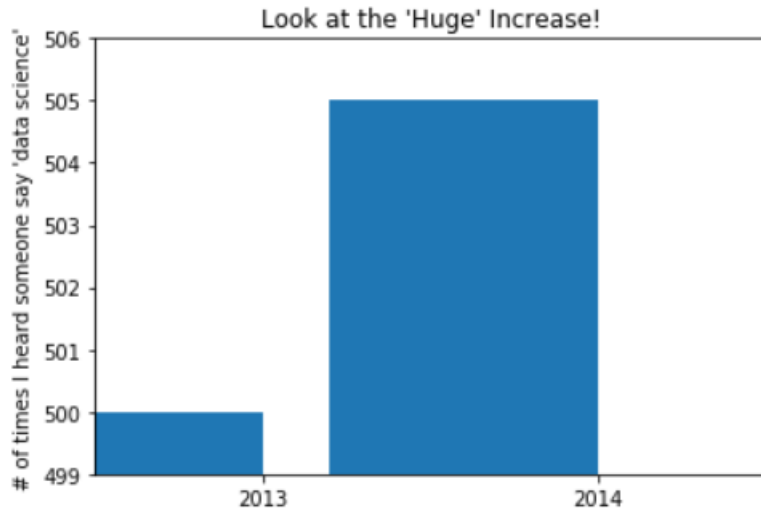


Um gráfico de barras é uma boa escolha quando precisamos mostrar como alguma quantidade varia entre algum conjunto discreto de itens

```
from matplotlib.ticker import FuncFormatter
x = np.arange(4)
money = [1.5e5, 2.5e6, 5.5e6, 2.0e7]
def millions(x, pos):
    'valor e posição'
    return '$%1.1fM' % (x * 1e-6)
formatter = FuncFormatter(millions)

fig, ax = plt.subplots()
ax.yaxis.set_major_formatter(formatter)
plt.bar(x, money)
plt.xticks(x, ('Bill', 'Fred', 'Mary', 'Sue'))
plt.show()
```

Temos que ser criteriosos ao usar `plt.axis()`. Ao criarmos um gráfico de barras **é considerado ruim iniciar o eixo y num ponto que não seja 0**, visto que é uma maneira fácil de enganar as pessoas. Observem:



Gráficos de barras são úteis para criar **histogramas**, que é a representação gráfica de uma distribuição de frequência. É uma estimativa da distribuição de probabilidade de uma variável contínua

```
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

As barras empilhadas são particularmente úteis quando desejamos **comparar elementos** de grupos distintos

```
# barra empilhada
N = 5
qtde_acoes = np.array([20, 15, 20, 15, 17])
qtde_fiis = np.array([15, 12, 14, 20, 15])
ind = np.arange(N)
width = 0.35      # Largura das barras

p1 = plt.bar(ind, qtde_acoes, width)
p2 = plt.bar(ind, qtde_fiis, width,
              bottom=qtde_acoes)

plt.ylabel('Quantidade de Ativos')
plt.title('Distribuição de Ativos na Carteira')
plt.xticks(ind, ('JAN', 'FEV', 'MAR', 'ABR', 'MAI'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Ações', 'FIIs'))

for x,y,val in zip(ind,qtde_acoes/2, qtde_acoes):
    plt.text(x, y, "%.1d"%val, ha="center", va="center")
for x,y,val in zip(ind,qtde_acoes+qtde_fiis/2, qtde_fiis):
    plt.text(x, y, "%.1d"%val, ha="center", va="center")

plt.show()
```

Um **scatterplot** é a escolha certa para **visualizar a relação entre dois conjuntos de dados**. A ilustração a seguir mostra a distribuição de pontos para homens e mulheres

```
pontos_f = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
pontos_m = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
range_pontos = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(range_pontos, pontos_f, color='r')
plt.scatter(range_pontos, pontos_m, color='g')
plt.xlabel('Range de Pontos')
plt.ylabel('Pontos')
plt.show()
```

É uma ferramenta gráfica para representar a variação de dados observados de uma variável numérica por meio de quartis.

São essencialmente úteis para detecção de outliers.

```
np.random.seed(19680801)

#criando dados fictícios
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```

Integrando Pandas, Matplotlib e Seaborn

Pandas oferece uma integração com matplotlib que facilita a criação de plots a partir do próprio Pandas.

O Seaborn, por sua vez, é construído sobre o matplotlib e simplifica a criação de muitos tipos de visualização.

Vamos começar com Pandas + Matplotlib:

Series e DataFrame do Pandas possuem um método *plot()* para criar alguns tipos básicos de plot. Por padrão, *plot()* cria um gráfico de linha:

```
s = pd.Series(np.random.standard_normal(10).cumsum(), index=np.arange(0, 100, 10))
s.plot()
```

```
df = pd.DataFrame(np.random.standard_normal((10, 4)).cumsum(0),
                  columns=["A", "B", "C", "D"],
                  index=np.arange(0, 100, 10))
plt.style.use('grayscale')
df.plot()
```

O atributo *style* informa o estilo, cor e marcador da linha a ser plotado. A tabela a seguir apresenta alguns atributos com suas respectivas descrições:

Argument	Description
label	Label for plot legend
ax	matplotlib subplot object to plot on; if nothing passed, uses active matplotlib subplot
style	Style string, like "ko- -", to be passed to matplotlib
alpha	The plot fill opacity (from 0 to 1)
kind	Can be "area", "bar", "barh", "density", "hist", "kde", "line", or "pie"; defaults to "line"
figsize	Size of the figure object to create
logx	Pass True for logarithmic scaling on the x axis; pass "sym" for symmetric logarithm that permits negative values
logy	Pass True for logarithmic scaling on the y axis; pass "sym" for symmetric logarithm that permits negative values
title	Title to use for the plot
use_index	Use the object index for tick labels
rot	Rotation of tick labels (0 through 360)
xticks	Values to use for x-axis ticks
yticks	Values to use for y-axis ticks
xlim	x-axis limits (e.g., [0, 10])
ylim	y-axis limits
grid	Display axis grid (off by default)

O objeto *plot* contém uma “família” de métodos de diferentes tipos de plots. Por exemplo, *df.plot()* é equivalente a *df.plot.line()*. Vamos ver como criar um gráfico de barras:

```
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.uniform(size=16), index=list("abcdefghijklmnop"))
data.plot.bar(ax=axes[0], color="black", alpha=0.7)
data.plot.barh(ax=axes[1], color="black", alpha=0.7)
```

Entretanto, matplotlib, mesmo integrado ao Pandas, possui uma série de críticas:

1. Sua API é relativamente low level, o que significa que é possível criar gráficos sofisticados, mas às custas de uma codificação mais exigente
2. Ele é mais de uma década mais velho que o Pandas, então a integração não é simples e geralmente você consegue trabalhar apenas com Series.

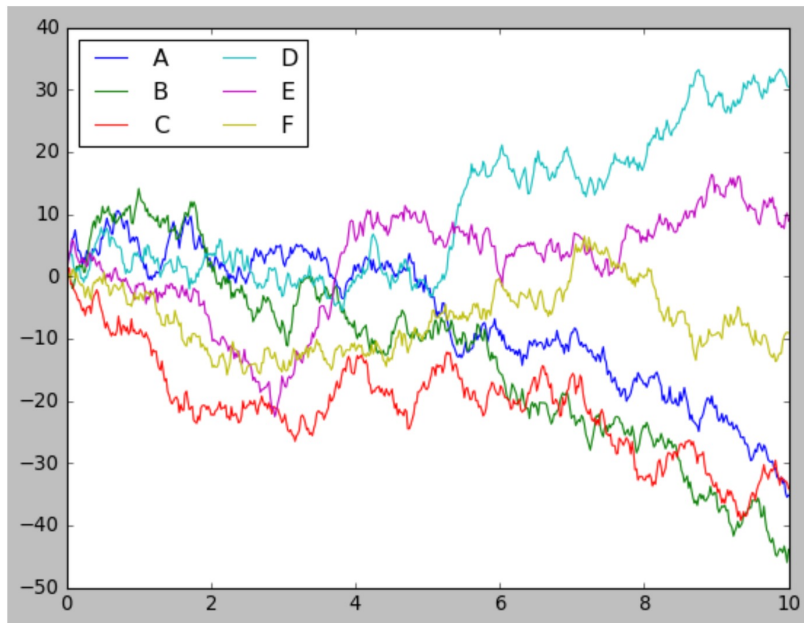
Uma das respostas a esses problemas é o Seaborn, que vamos entender como usá-lo para deixar o matplotlib mais bonito.

Seaborn fornece uma API construída sobre o Matplotlib que oferece opções sensatas para estilos de plots e padrões de cores; define funções simples de alto nível para tipos de plots estatísticos comuns e integra-se com a funcionalidade fornecida pelo Pandas DataFrames.

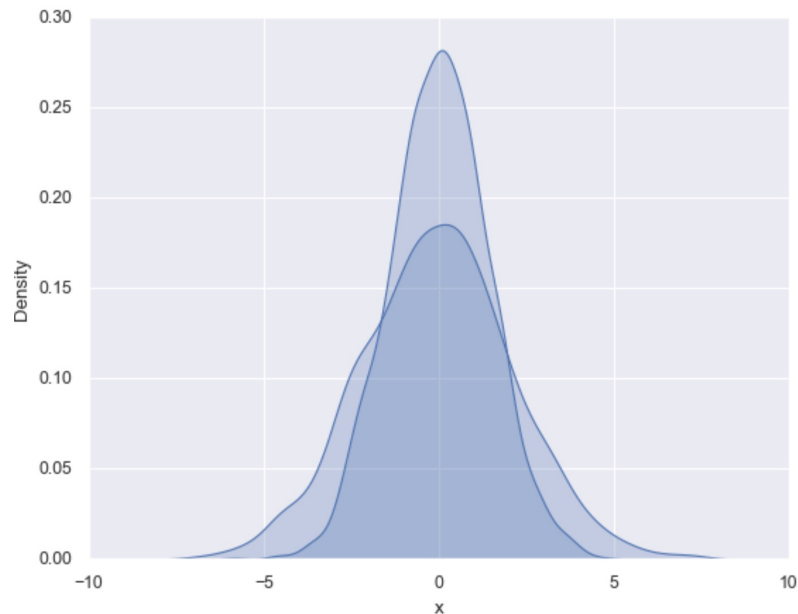
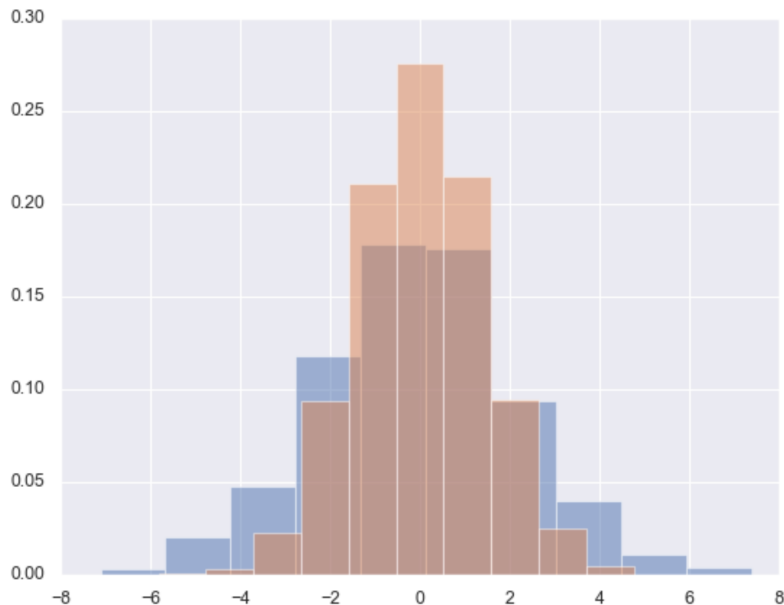
Nota 1: o matplotlib tem endereçado esses problemas e sua versão 2.0 é bastante adequada.

Nota 2: o intuito aqui não é entender a fundo o Seaborn, mas como este se integra ao matplotlib para deixar suas visualizações melhores.

Observe os dois gráficos abaixo. Ambos foram construídos com matplotlib, mas o da direita teve sua aparência melhorada com o seaborn:

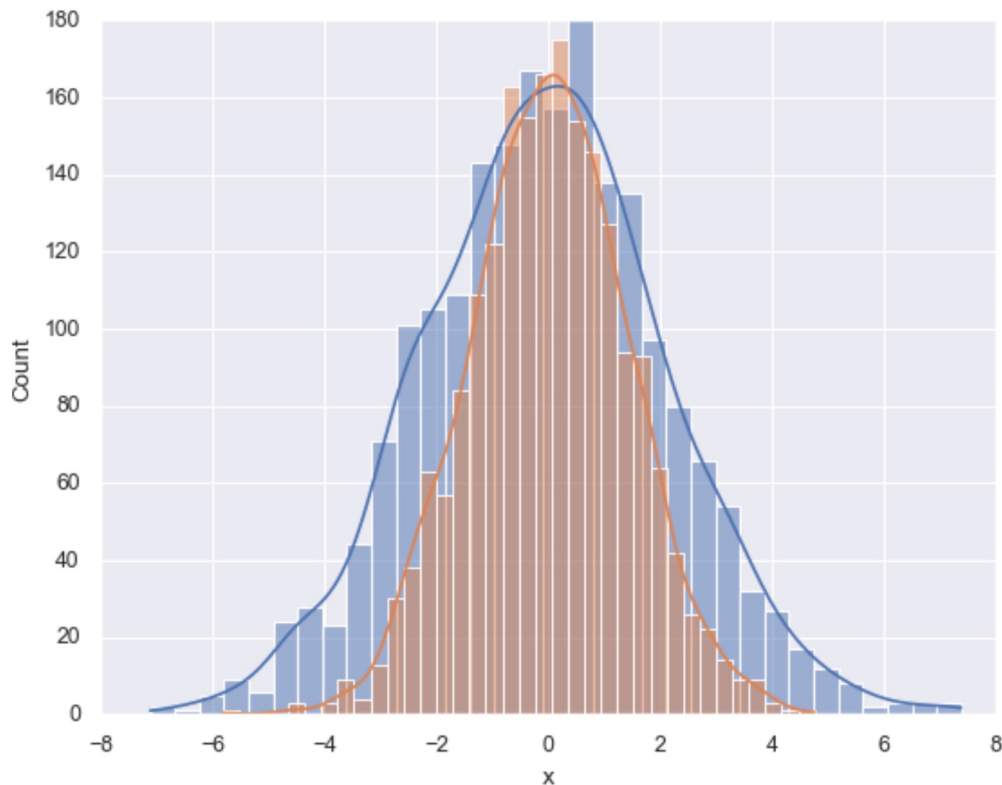


Aqui está uma comparação entre histogramas usando matplotlib e seaborn:



Ao invés de um histograma, podemos uma estimativa da distribuição usando *kernel density estimation*. Observe o código:

E é possível combinar histograma e o KDE (*kernel density estimation*):

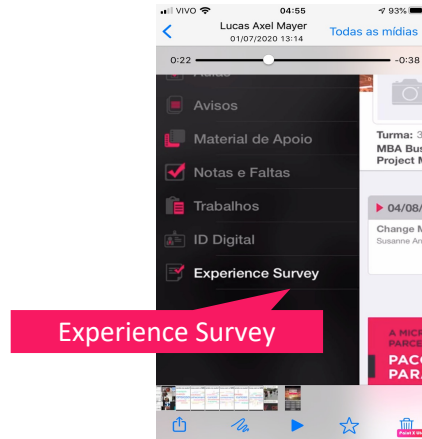


Por fim, um tipo de gráfico que é particularmente útil para explorar correlações entre dados multidimensionais é o *pair plot*. Veja o exemplo no notebook

Resolver o exercício no notebook

O que você achou da aula de hoje?


Entrar no aplicativo FIAPP, e no menu clicar em Experience Survey



Ou pelo link: <https://fiap.me/Pesquisa-MBA>

Obrigado!

profdheny.fernandes@fiap.com.br

 /dhenyfernandes

FIAP MBA⁺

Copyright © 2018 | Professor Dheny R. Fernandes

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

FIAP