UNIDADE 3
Infraestrutura para Manipulação e Visualização de Dados

Disciplina: Tópicos Especiais II (DEC7552)

Prof. Alexandre L. Gonçalves

E-mail: a.l.goncalves@ufsc.br

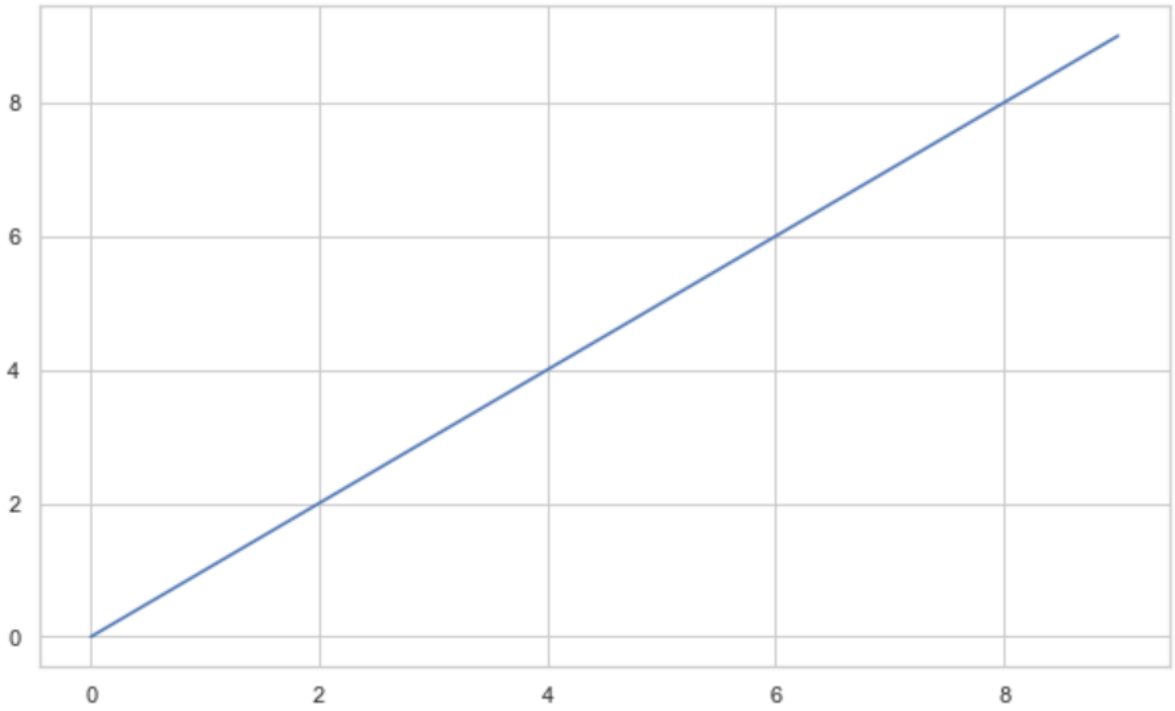
Plotagem e Visualização

- Gerar visualizações informativas (também chamada de plotagens) é uma tarefa importante em análise de dados;
- Python possui várias bibliotecas para criar visualizações estáticas ou dinâmicas. Um das mais utilizadas é a biblioteca matplotlib;
- A matplotlib é uma biblioteca criada para plotagens com qualidade para publicação (geralmente, bidimensionais);
- O projeto foi criado em 2002 por John Hunter visando possibilitar uma interface de plotagem do tipo MATLAB em Python.

Plotagem e Visualização

- Plotagem simples utilizando uma sequência.

```
import matplotlib.pyplot as plt

import numpy as np
data = np.arange(10)
print(data)
plt.plot(data)
```

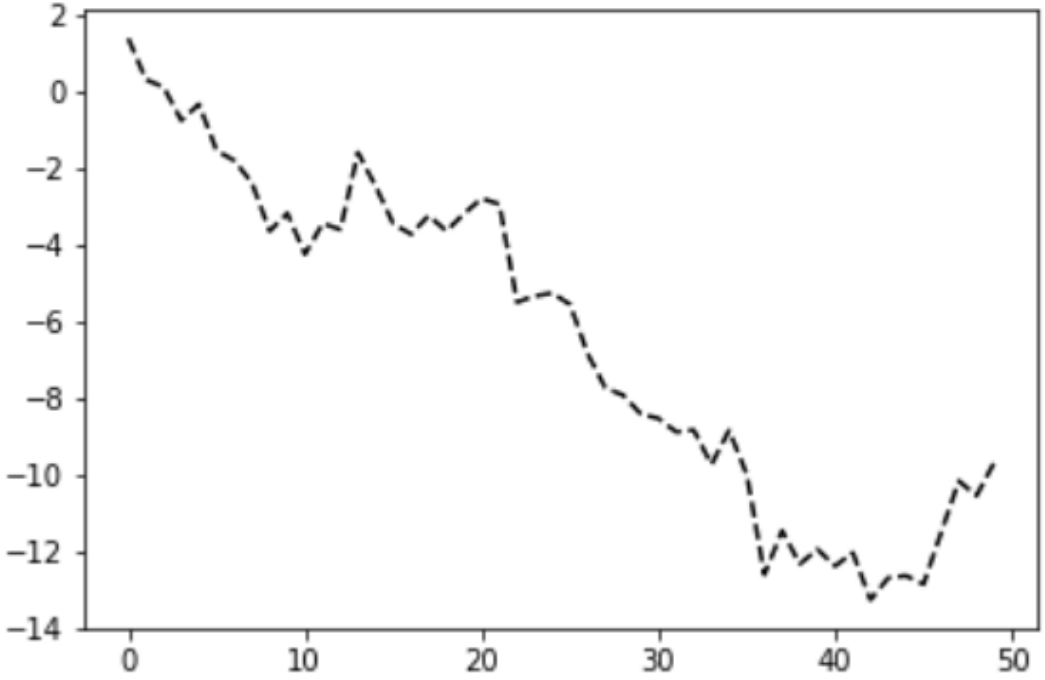


Figuras e Subplotagens

- As plotagens na biblioteca matplotlib ficam em um objeto Figure.
- No IPython, uma janela de plotagem vazia aparecerá, mas, no Jupyter, nada será exibido até que mais alguns comandos sejam usados;
- Para criar uma plotagem é necessário utilizar o método plot() informando o conteúdo e a opção de estilo da linha;
- k-- é uma opção de estilo que informa que uma linha tracejada será utilizada na plotagem.

Figuras e Subplotagens

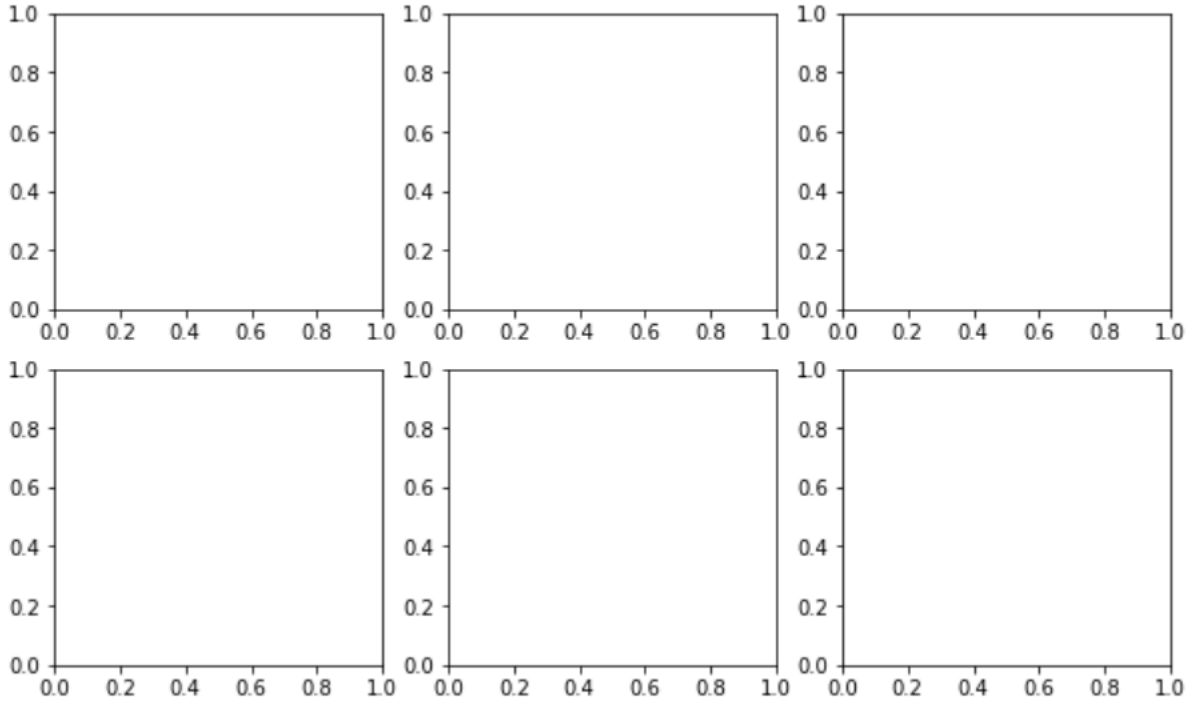
```
plt.plot(np.random.randn(50).cumsum(), 'k--')
```



Figuras e Subplotagens

- Criar uma figura com uma grade de subplotagens é uma tarefa bem comum;
- Isso é realizado através do método plt.subplots().

```
fig, axes = plt.subplots(2, 3)
```

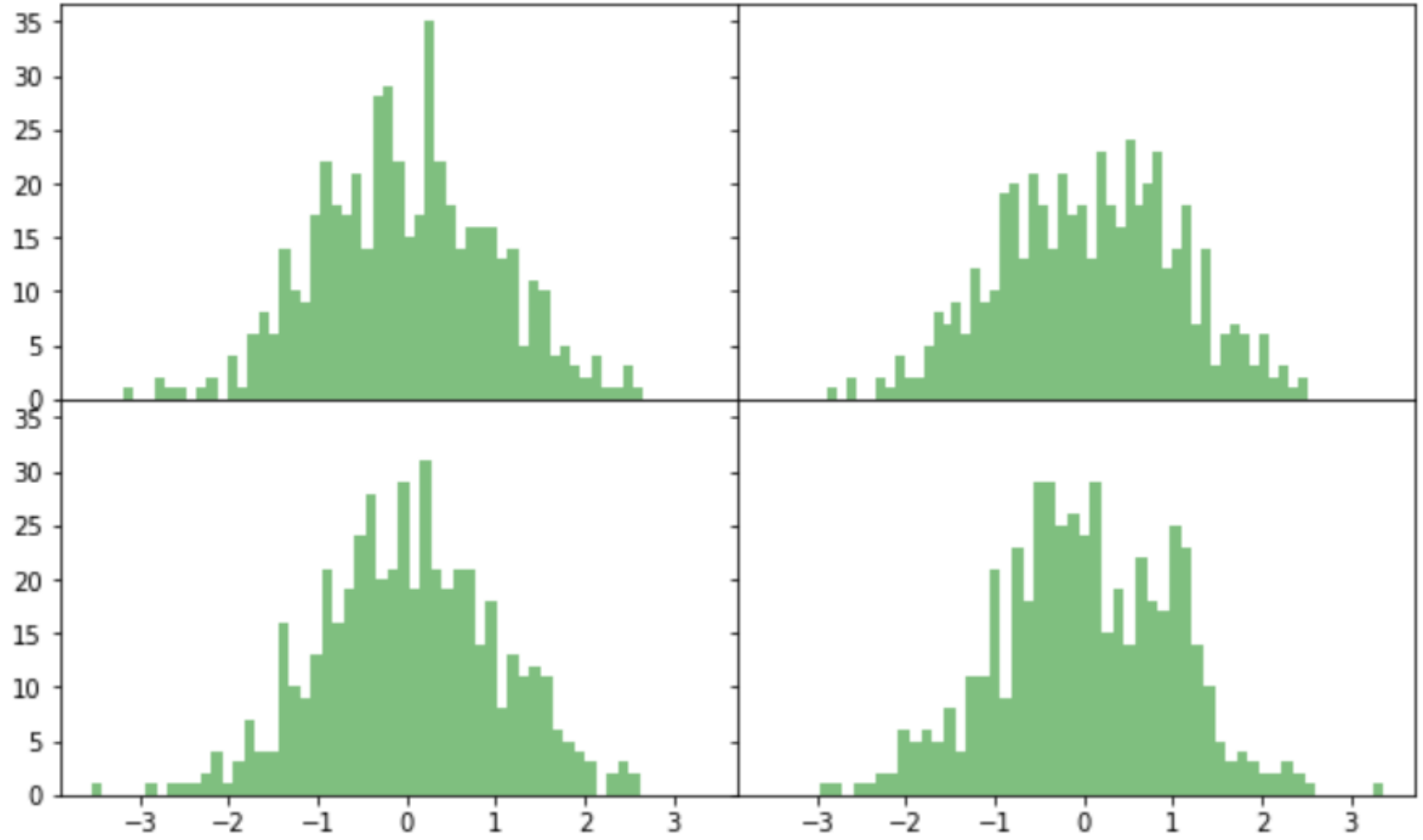


Ajustando o Espaçamento em torno das Subplotagens

- Por padrão, a matplotlib deixa determinado espaço para preenchimento em torno da subplotagens;
- O espaçamento pode ser alterado usando o método subplots_adjust dos objetos Figure.

```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='g', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
```

- Ajustando o Espaçamento em torno das Subplotagens



Cores, Marcadores e Estilos de Linha

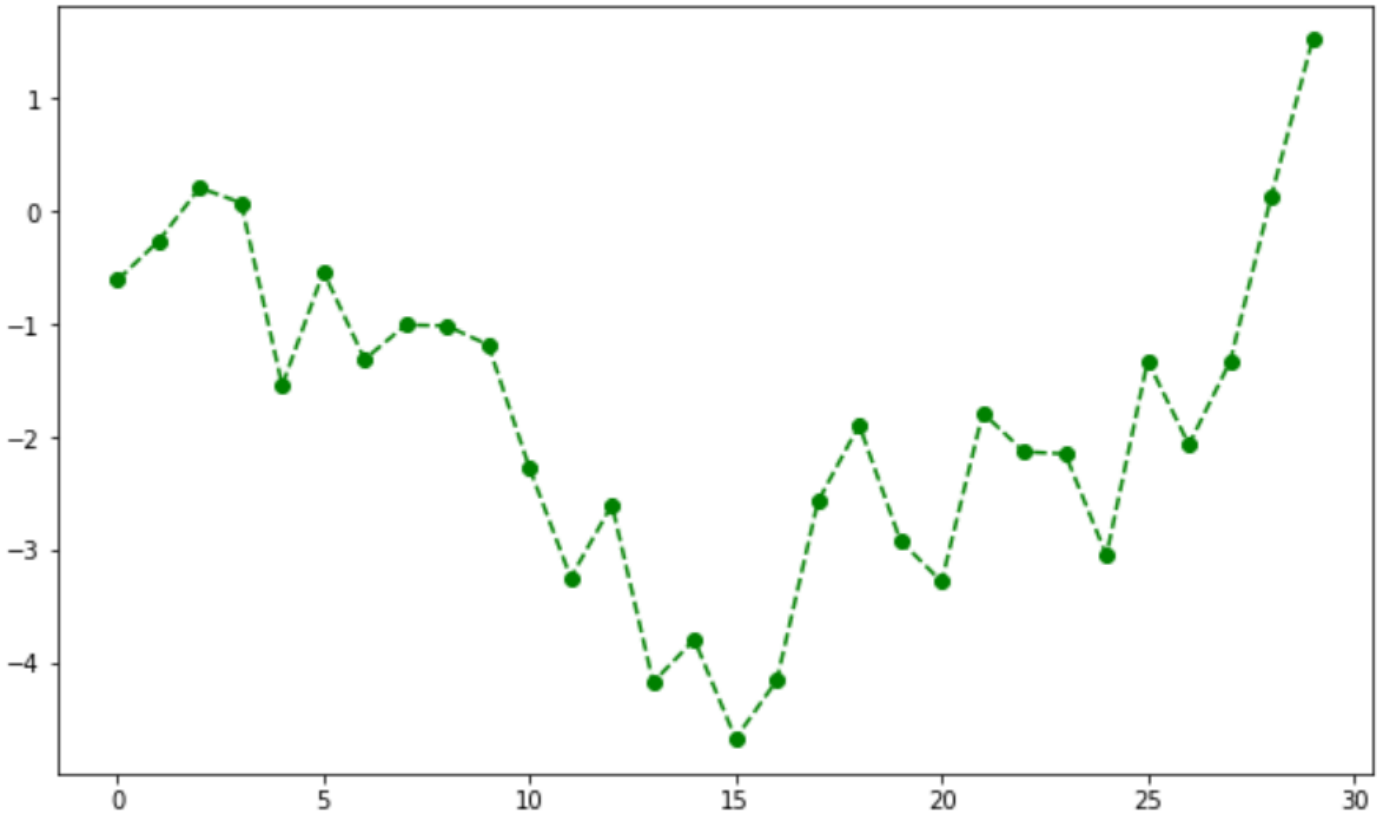
- O método principal *plot* aceita *arrays* de coordenadas *x* e *y*, e uma *string* abreviada informando a cor e o estilo da linha;

plt.plot(x, y, ‘g--’)

- Outra possibilidade é a utilização das propriedades *color*, *linestyle* e *marker*.

```
from numpy.random import randn
plt.plot(randn(30).cumsum(), linestyle='dashed', color='g', marker='o')
```

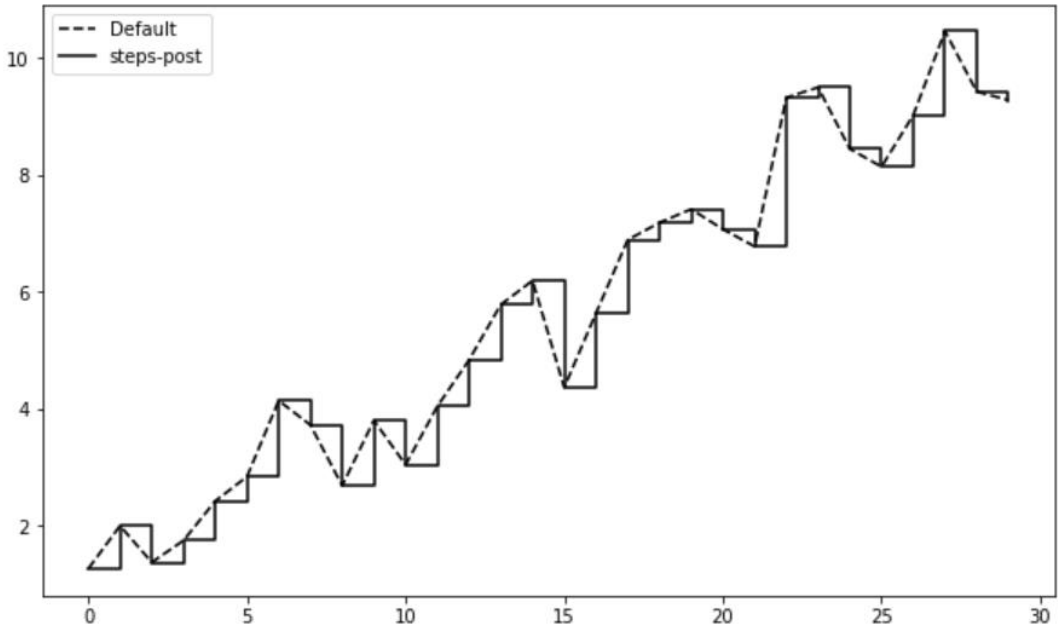
- Cores, Marcadores e Estilos de Linha



Cores, Marcadores e Estilos de Linha

- É possível também realizar uma interpolação dos dados.

```
data = np.random.randn(30).cumsum()
plt.plot(data, 'k--', label='Default')
plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')
```



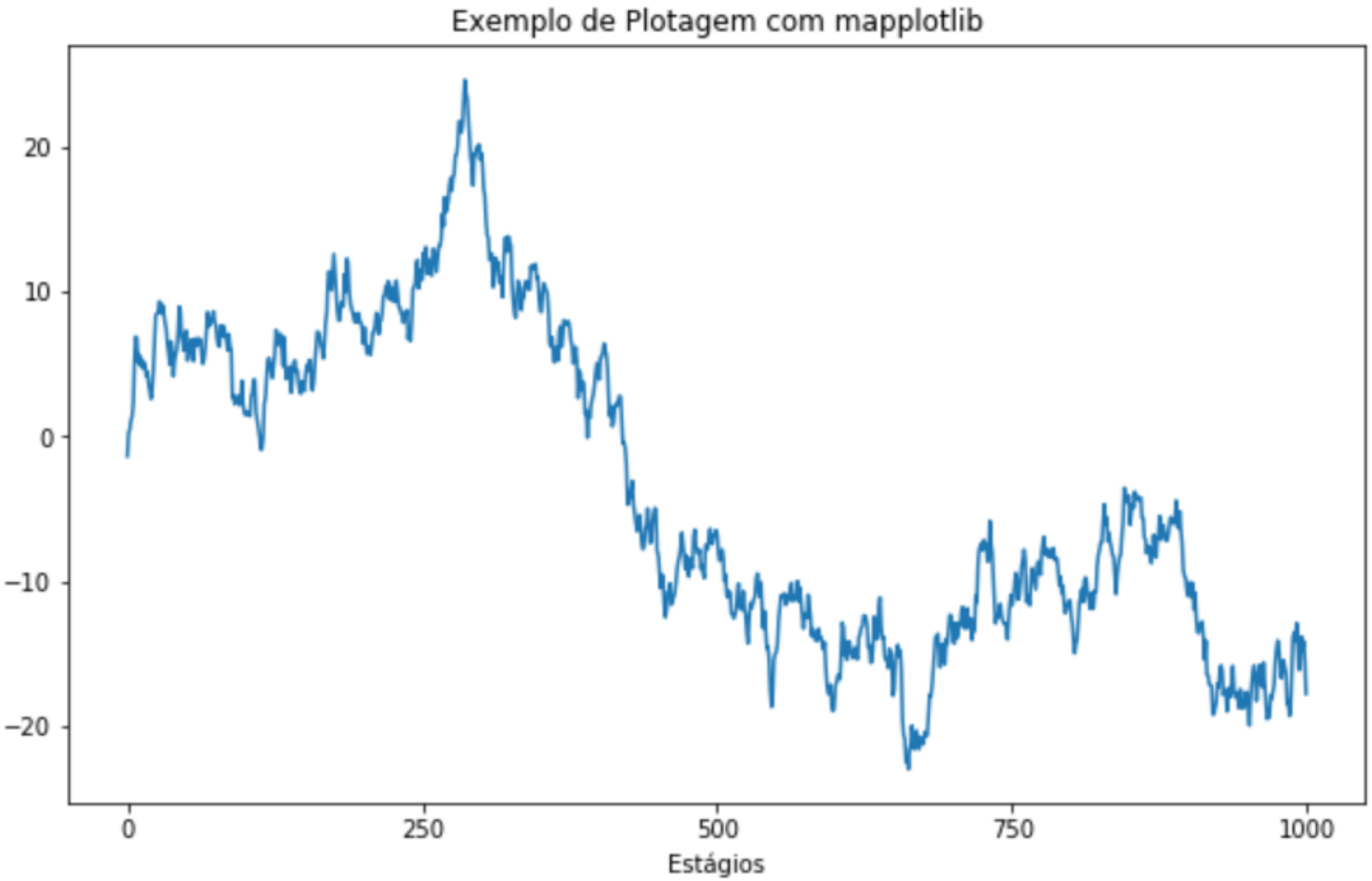
Tiques, Rótulos e Legendas

- Para a maioria dos tipos de decoração das plotagens pode-se utilizar a interface pyplot da matplotlib;
- A interface pyplot é constituída de métodos como xlim, xticks e xticklabels. Eles controlam o intervalo da plotagem, as localizações dos tiques e os seus rótulos, respectivamente;
- Para alterar os tiques do eixo x, a opção mais simples é usar set_xticks e set_xticklabels. O primeiro método instrui acerca do local para posicionar os tiques, enquanto o segundo define os rótulos dos tiques.

Tiques, Rótulos e Legendas

```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['um', 'dois', 'três', 'quatro', 'cinco'],
                            rotation=30, fontsize='small')

ax.set_title('Exemplo de Plotagem com mapplotlib')
ax.set_xlabel('Estágios')
ax.plot(np.random.randn(1000).cumsum())
```

Tiques, Rótulos e Legendas



Acrescentando Legendas

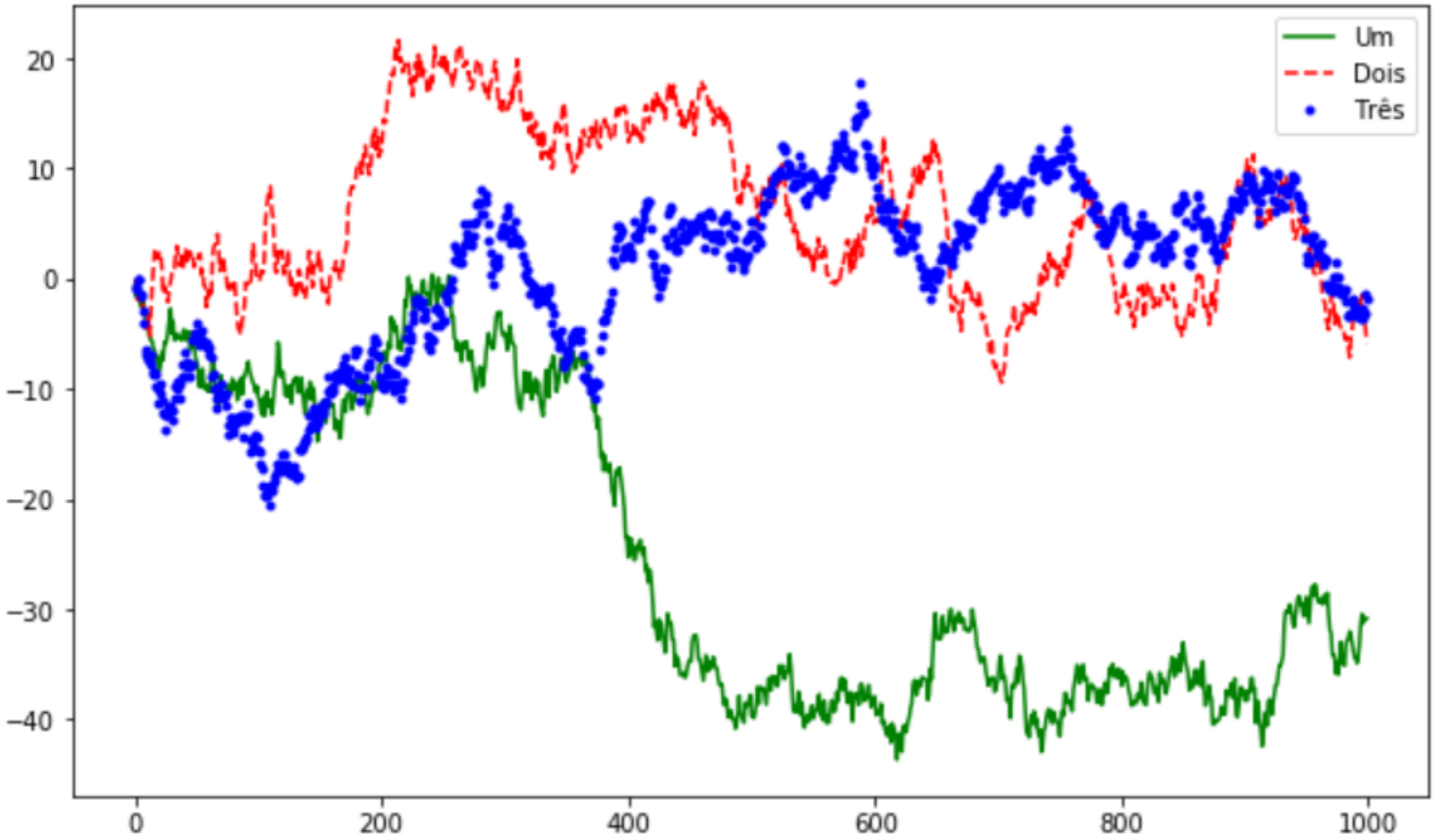
- As legendas constituem-se em um recurso importante para identificar elementos da plotagem;
- O modo mais simples é passar o argumento **label** ao adicionar cada parte da plotagem;
- Após isso, pode-se chamar o método legend().

Acrescentando Legendas

- As legendas constituem-se em um recurso importante para identificar elementos da plotagem;
- O modo mais simples é passar o argumento **label** ao adicionar cada parte da plotagem;
- Após isso, pode-se chamar o método legend().

```
from numpy.random import randn
fig = plt.figure();
ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'k', color='g', label='Um')
ax.plot(randn(1000).cumsum(), 'k--', color='r', label='Dois')
ax.plot(randn(1000).cumsum(), 'k.', color='b', label='Três')
ax.legend(loc='best')
```

Acrescentando Legendas



Anotações em uma Subplotagem

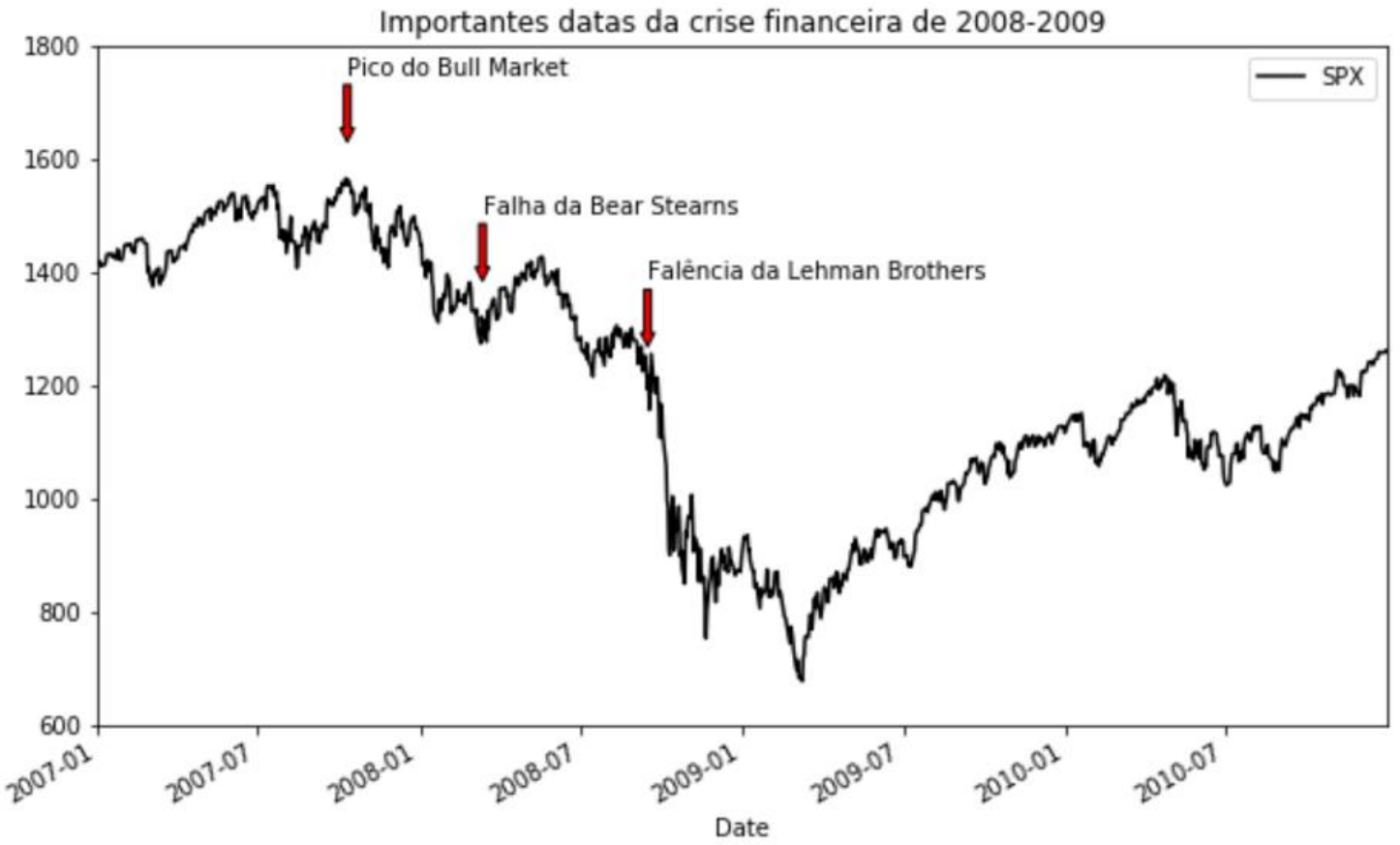
- Além dos tipos padrões de plotagem, talvez seja necessário desenhar determinas anotações na plotagem;

- É possível adicionar anotações e textos usando as funções *text*, *arrow* e *annotate*.

Anotações em uma Subplotagem

```
from datetime import datetime

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

data = pd.read_csv('spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']

spx.plot(ax=ax, style='k-')

crisis_data = [
    (datetime(2007, 10, 11), 'Pico do Bull Market'),
    (datetime(2008, 3, 12), 'Falha da Bear Stearns'),
    (datetime(2008, 9, 15), 'Falência da Lehman Brothers')
]

for date, label in crisis_data:
    ax.annotate(label, xy=(date, spx.asof(date) + 75),
                xytext=(date, spx.asof(date) + 225),
                arrowprops=dict(facecolor='black', headwidth=4, width=2,
                                headlength=4),
                horizontalalignment='left', verticalalignment='top')

# Zoom in on 2007-2010
ax.set_xlim(['1/1/2007', '31/12/2010'])
ax.set_ylim([600, 1800])

ax.set_title('Importantes datas da crises financeira de 2008-2009')
```

Anotações em uma Subplotagem



Salvando Plotagens em Arquivos

- Pode-se salvar a figura ativa usando plt.savefig.

```
plt.savefig('plot.png')

plt.savefig('plot.png', dpi=400, bbox_inches='tight')
```

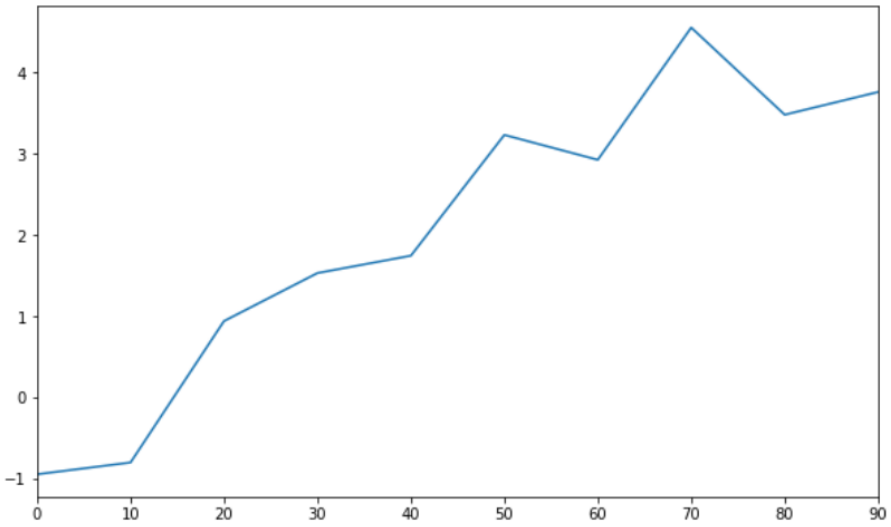
Plotagem com Pandas e Seaborn

- Na biblioteca Pandas pode-se ter várias colunas de dados, junto com rótulos para linhas e colunas;
- Existem métodos que simplificam a criação de visualizações a partir de objetos Series e DataFrames;
- Outra biblioteca é a Seaborn, uma biblioteca gráfica que simplifica a criação de vários tipos comuns de visualização.

Plotagem de Linhas

- Tanto Series quanto DataFrame têm um atributo plot para criar alguns tipos básicos de plotagem;
- Por padrão, plot() cria plotagens de linha.

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

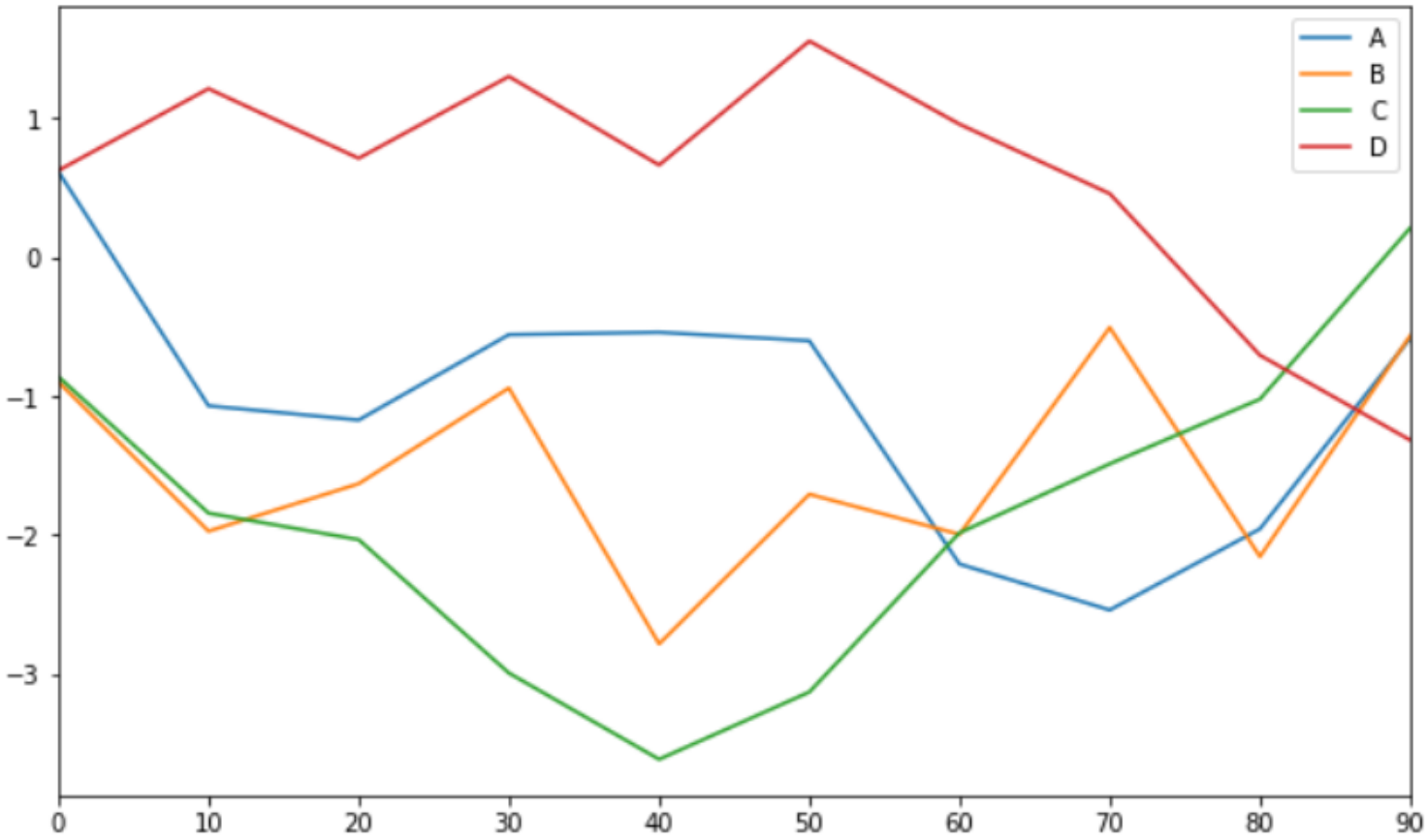


Plotagem de Linhas

- A maioria dos métodos de plotagem da biblioteca Pandas aceita um parâmetro ax opcional, que pode ser um objeto de subplotagem da matplotlib;
- O método plot de DataFrame plota cada uma de suas colunas como uma linha diferente na mesma subplotagem, criando uma legenda automaticamente.

```
df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                  columns=['A', 'B', 'C', 'D'],
                  index=np.arange(0, 100, 10))
df.plot()
```

Plotagem de Linhas

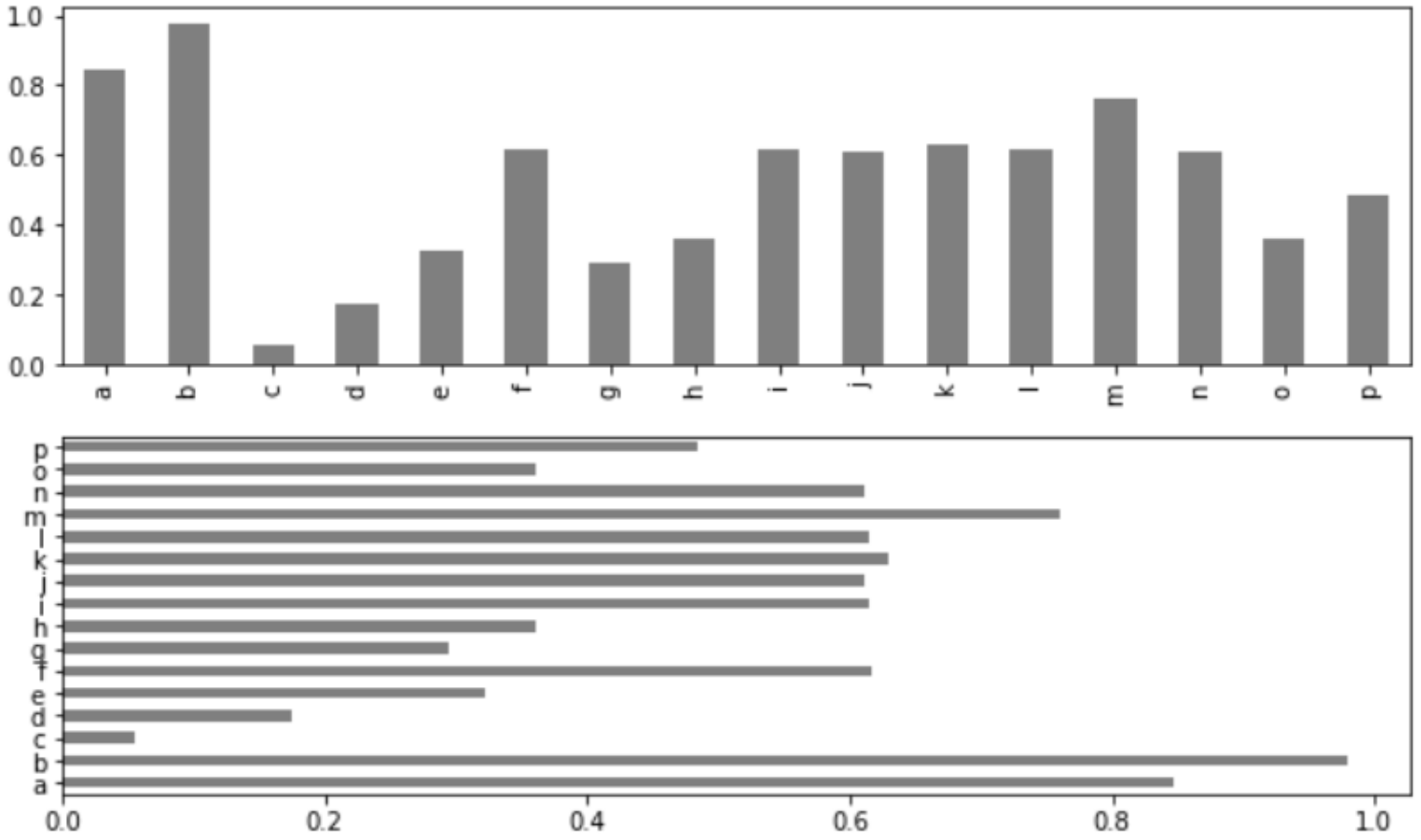


Plotagem de Barras

- plot.bar() e plot.barh() criam plotagens de barra vertical e horizontal;
- Neste caso, o índice de Series ou de DataFrame será utilizado como os tiques de x (bar) ou de y (barh).

```
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.5)
data.plot.barh(ax=axes[1], color='k', alpha=0.5)
```

Plotagem de Barras



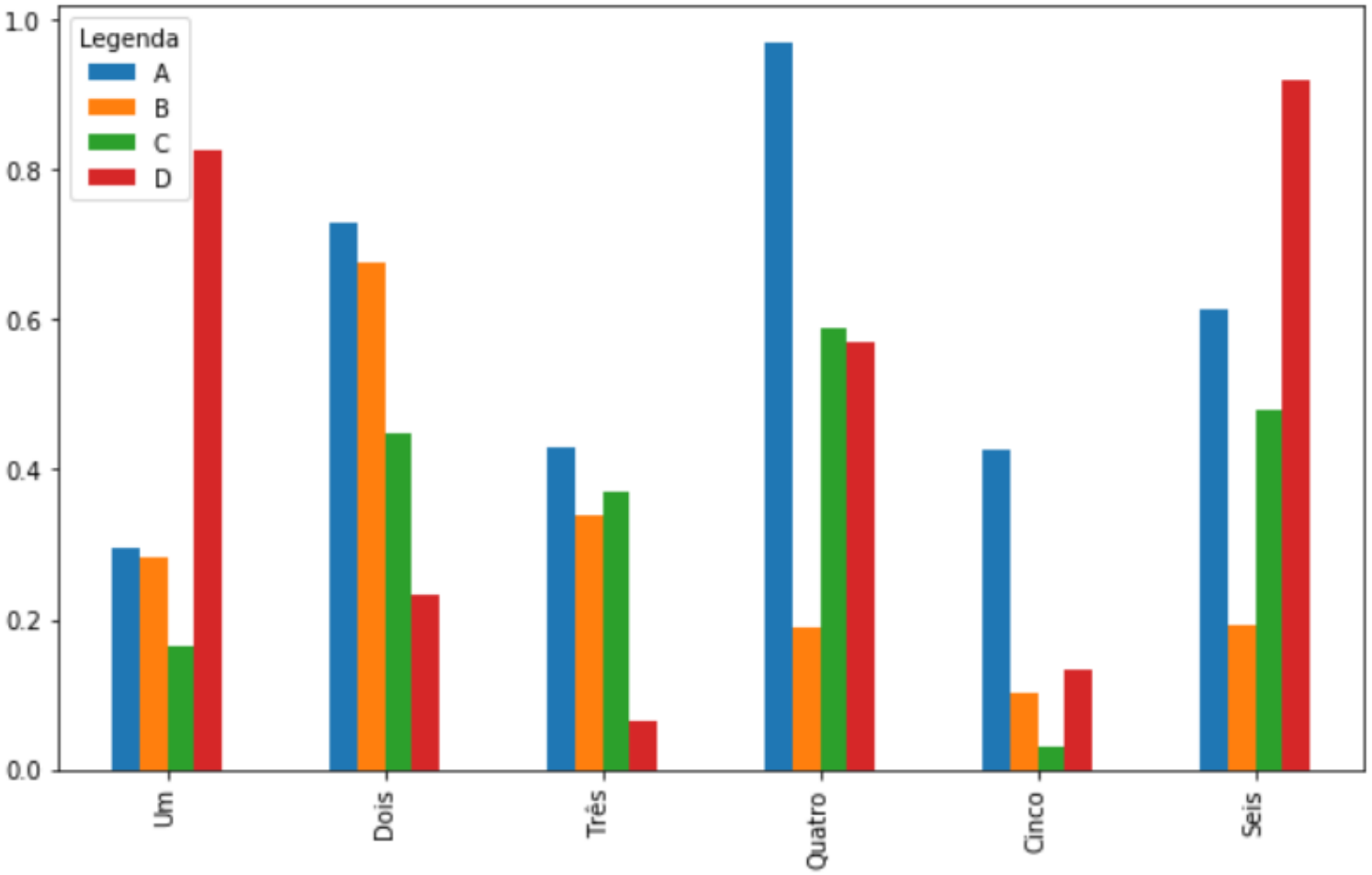
Plotagem de Barras

- As opções *color* e *alpha* definem a cor das plotagens e o uso de transparência parcial para o preenchimento;

- Com um DataFrame, as plotagens de barra agrupam os valores de cada linha em um grupo de barras, lado a lado, para cada valor.

```
df = pd.DataFrame(np.random.rand(6, 4),
                  index=['Um', 'Dois', 'Três', 'Quatro', 'Cinco', 'Seis'],
                  columns=pd.Index(['A', 'B', 'C', 'D'], name='Legenda'))
df
df.plot.bar()
```

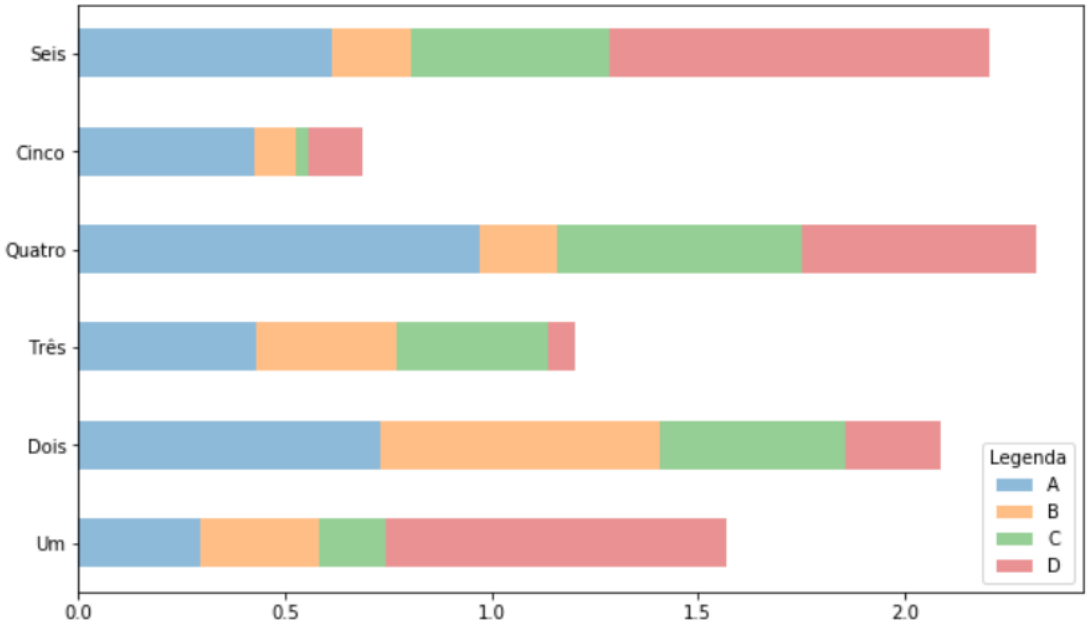
Plotagem de Barras



Plotagem de Barras

- É possível criar uma plotagem de barras empilhadas a partir de um DataFrame utilizando a propriedade *stacked=True* do método *barh*.

df.plot.barh(stacked=True, alpha=0.5)

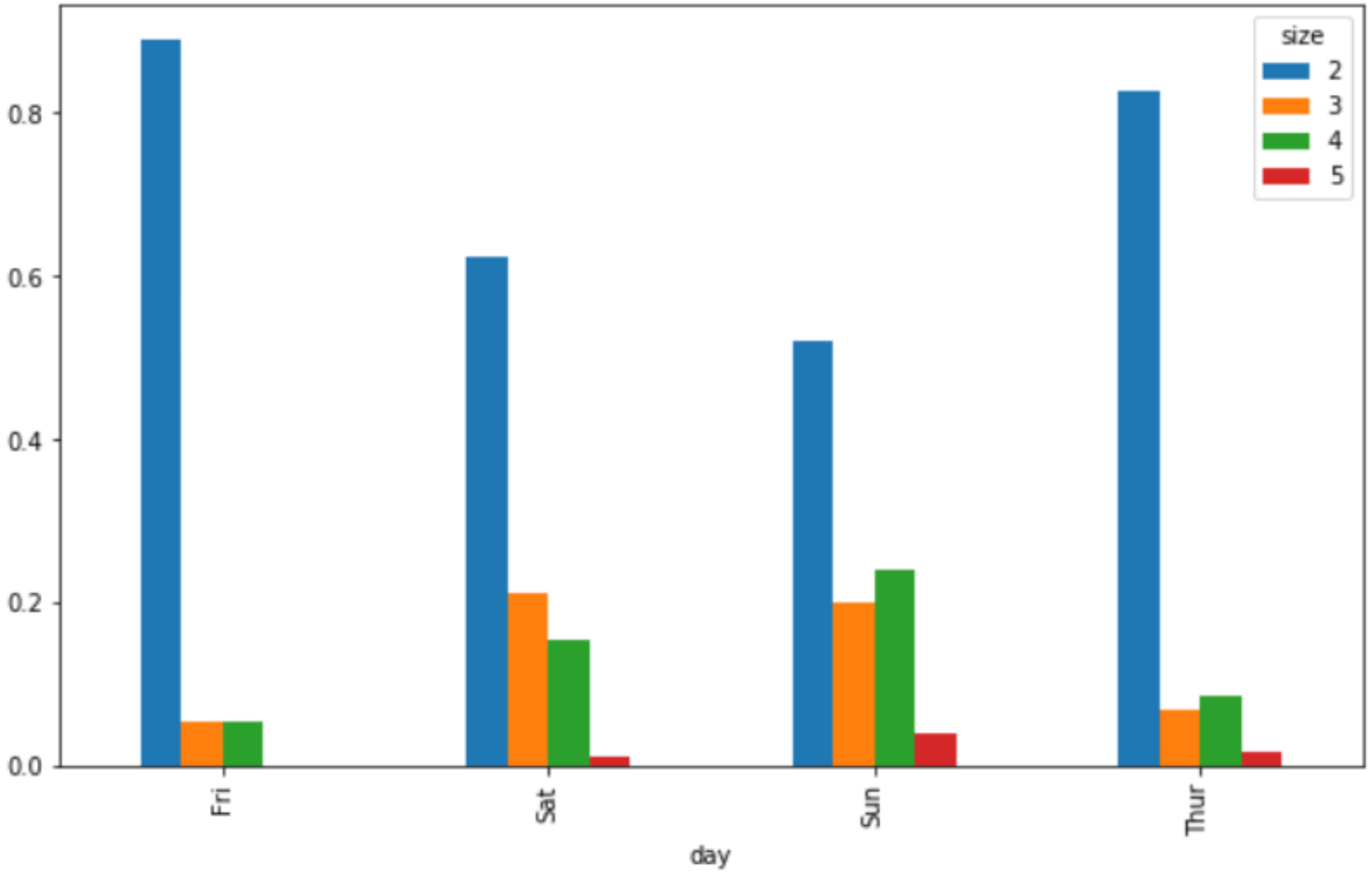


Plotagem de Barras

- Usando o conjunto de dados de gorjetas, seria possível gerar uma plotagem de barras empilhadas que exibisse o percentual de pontos de dados para cada tamanho de grupo a cada dia.

```
tips = pd.read_csv('tips.csv')
party_counts = pd.crosstab(tips['day'], tips['size'])
party_counts
print(party_counts)
party_counts = party_counts.loc[:, 2:5]
print(party_counts)

# Normaliza a soma entre 0 e 1
party_pcts = party_counts.div(party_counts.sum(1), axis=0)
party_pcts
party_pcts.plot.bar()
```

Plotagem de Barras

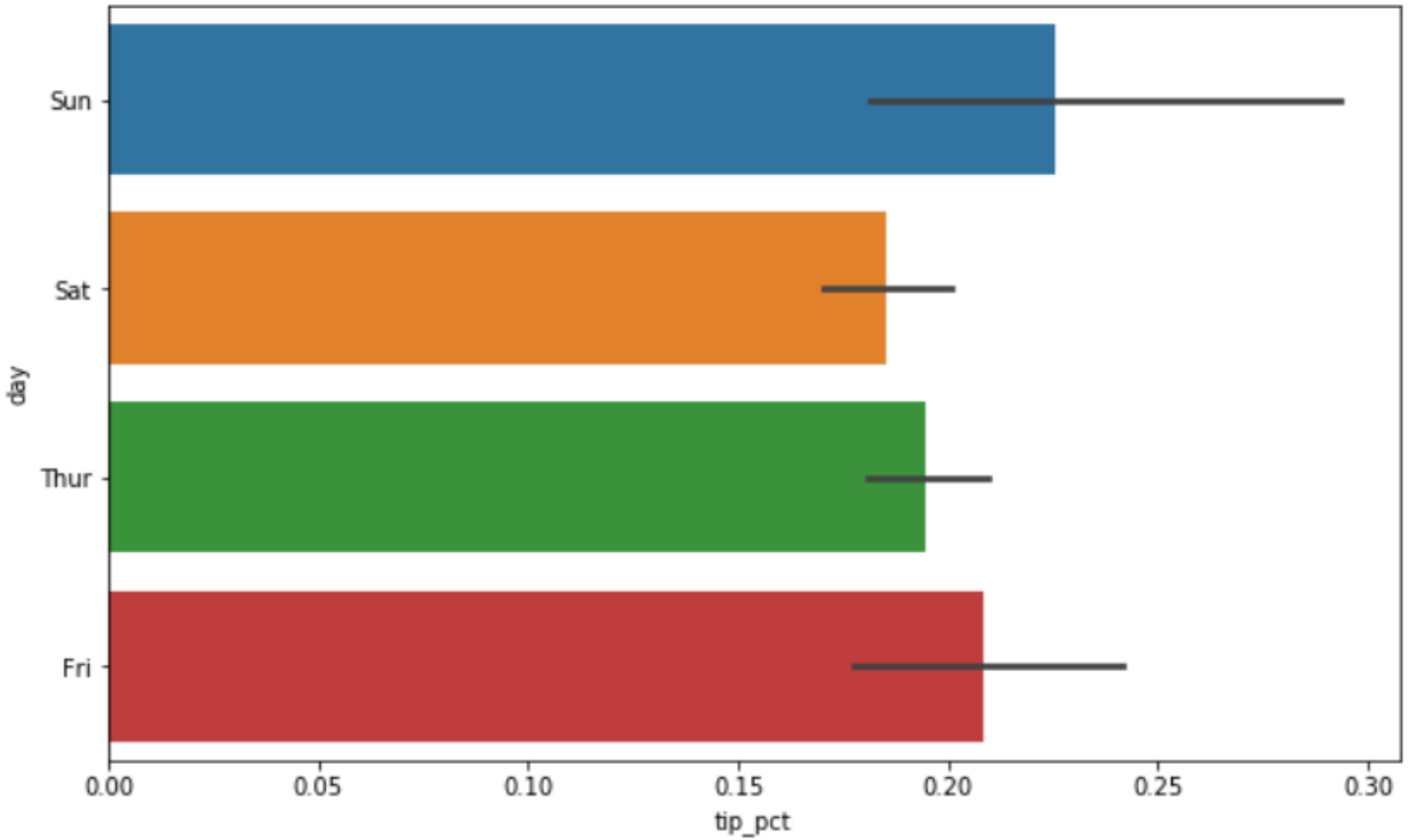


Plotagem de Barras

- Com dados que exijam agregação ou resumo antes da criação de uma plotagem, a utilização da biblioteca Seaborn pode simplificar bastante;
- As linhas pretas desenhadas nas barras representam o intervalo de confiança de 95%;
- O exemplo a seguir apresenta o percentual médio de gorjetas por dia.

```
import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
print(tips.head())
sns.barplot(x='tip_pct', y='day', data=tips, orient='h')
```

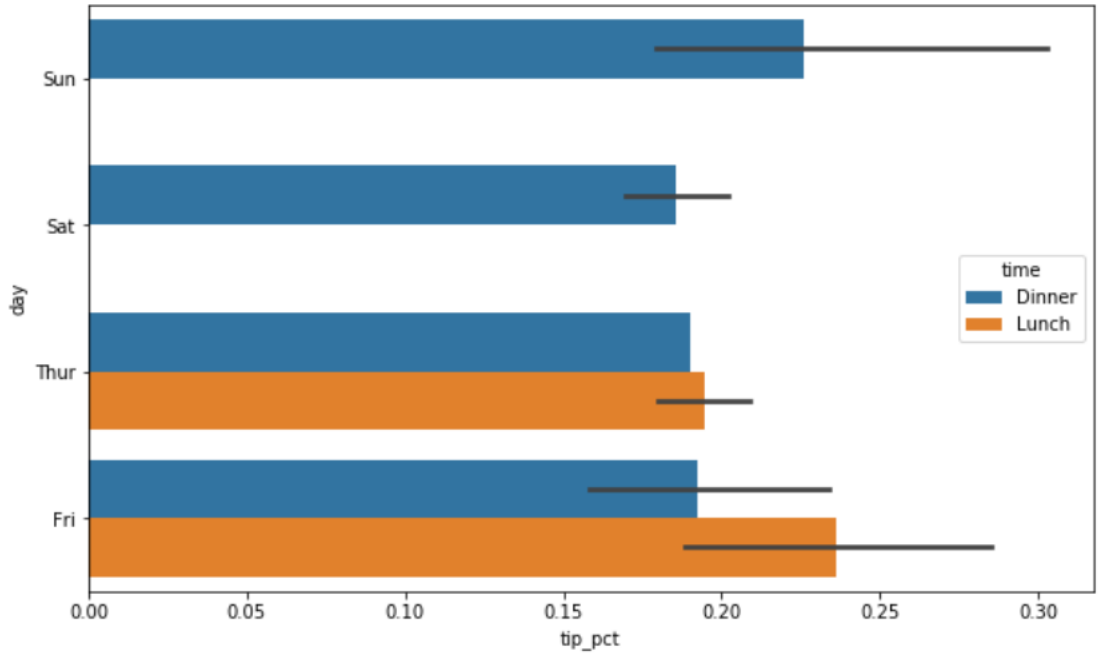
Plotagem de Barras



Plotagem de Barras

- Através da propriedade *hue* é possível separar determinada barra utilizando um valor adicional de categoria.

```
sns.barplot(x='tip_pct', y='day', hue='time', data=tips, orient='h')
```

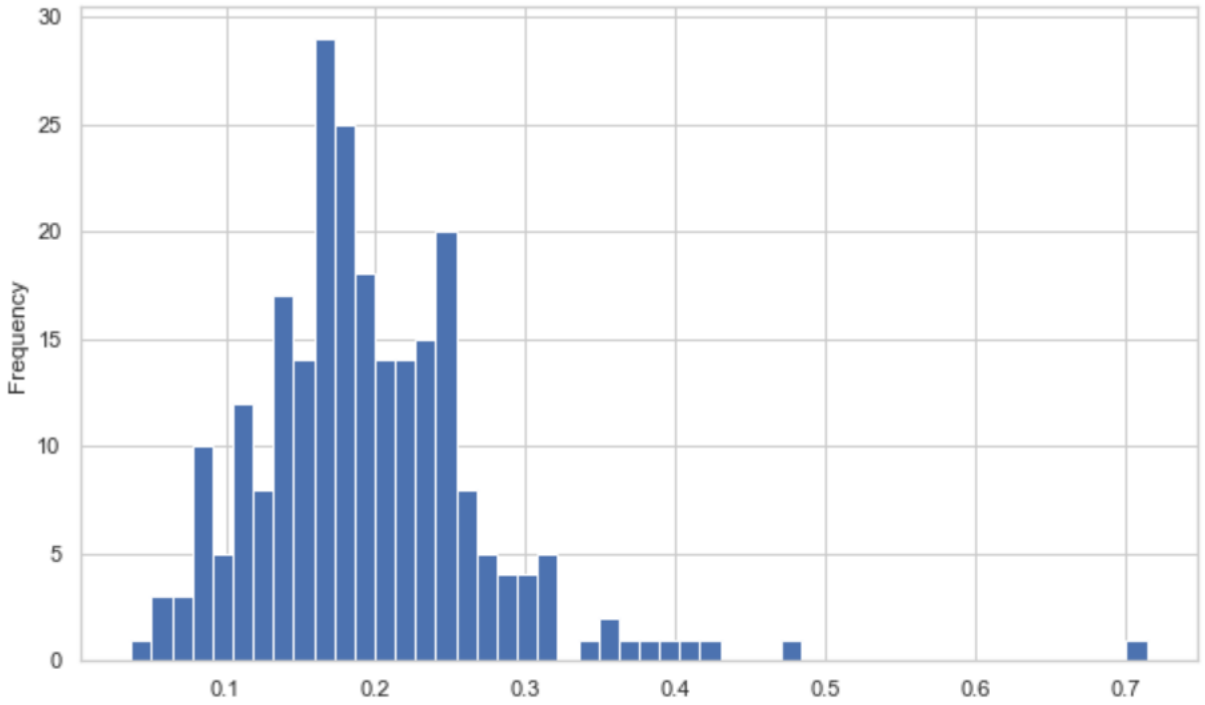


Histogramas e Plotagens de Densidade

- Um histograma é um tipo de plotagem de barras que oferece uma exibição discreta das frequências dos valores;
- Os pontos são separados em compartimentos (bins) discretos, uniformemente espaçados, permitindo a plotagem dos dados nesses compartimentos;
- Utilizando os dados de gorjetas pode-se gerar um histograma das porcentagens de gorjetas sobre o total das contas através do método plot.hist().

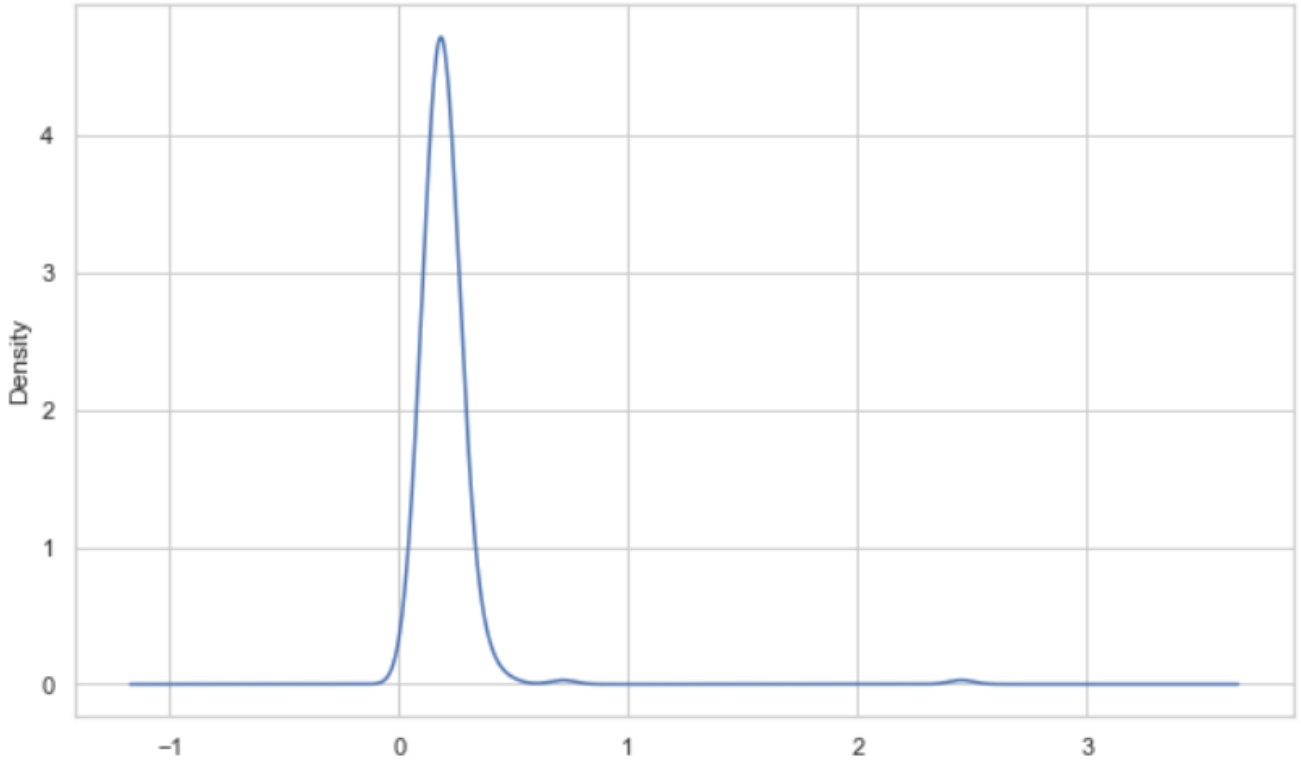
Histogramas e Plotagens de Densidade

```
print(tips.head())
tips['tip_pct'].plot.hist(bins=50)
```



- Histogramas e Plotagens de Densidade

  - O uso de plot.density gera uma plotagem de densidade que utiliza a estimativa convencional de distribuição normal.

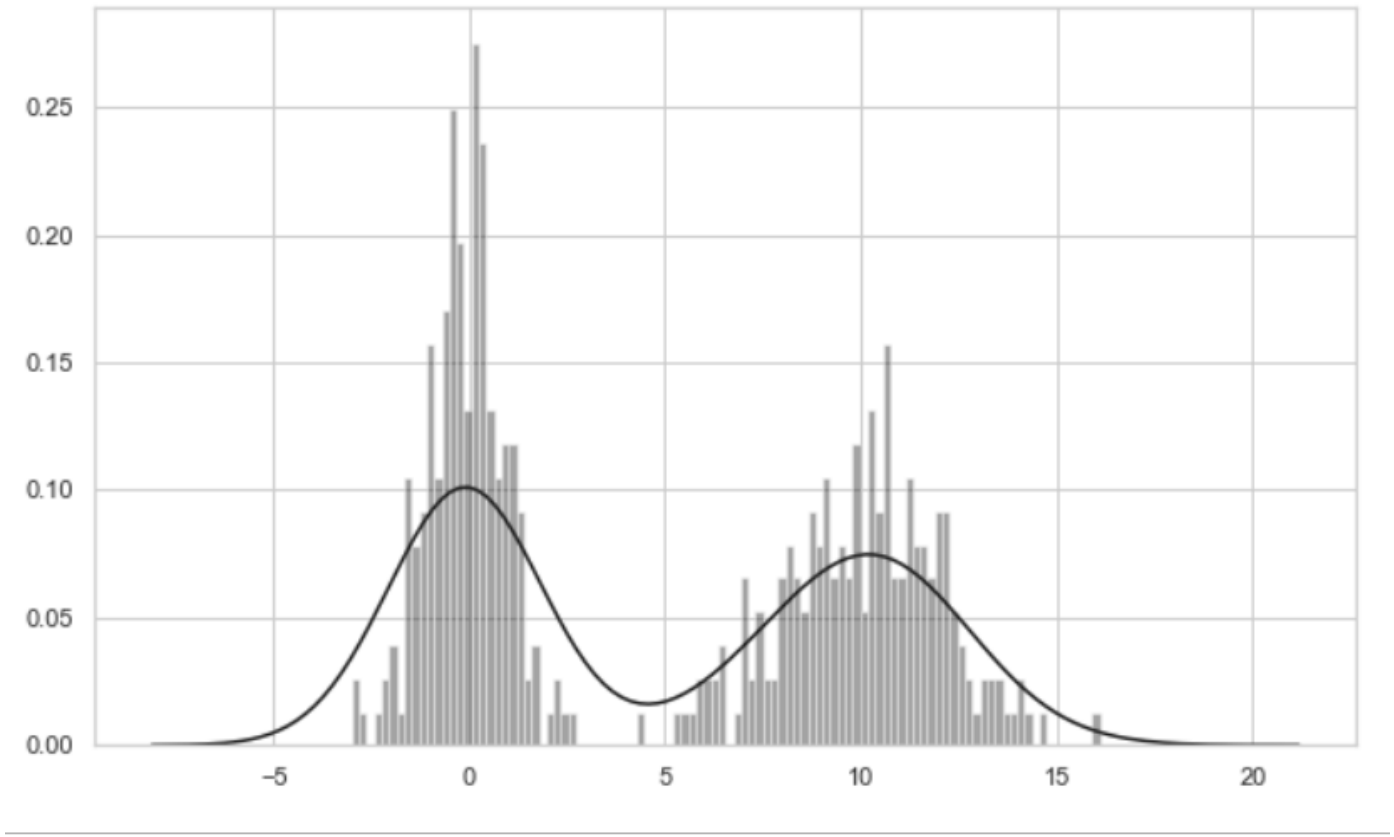


Histogramas e Plotagens de Densidade

- A biblioteca Seaborn gera histograma e plotagens de densidade mais facilmente através do método distplot;
- Este método é capaz de plotar tanto um histograma quanto uma estimativa de densidade contínua simultaneamente.

```
comp1 = np.random.normal(0, 1, size=200)
comp2 = np.random.normal(10, 2, size=200)
values = pd.Series(np.concatenate([comp1, comp2]))
sns.distplot(values, bins=100, color='k')
```

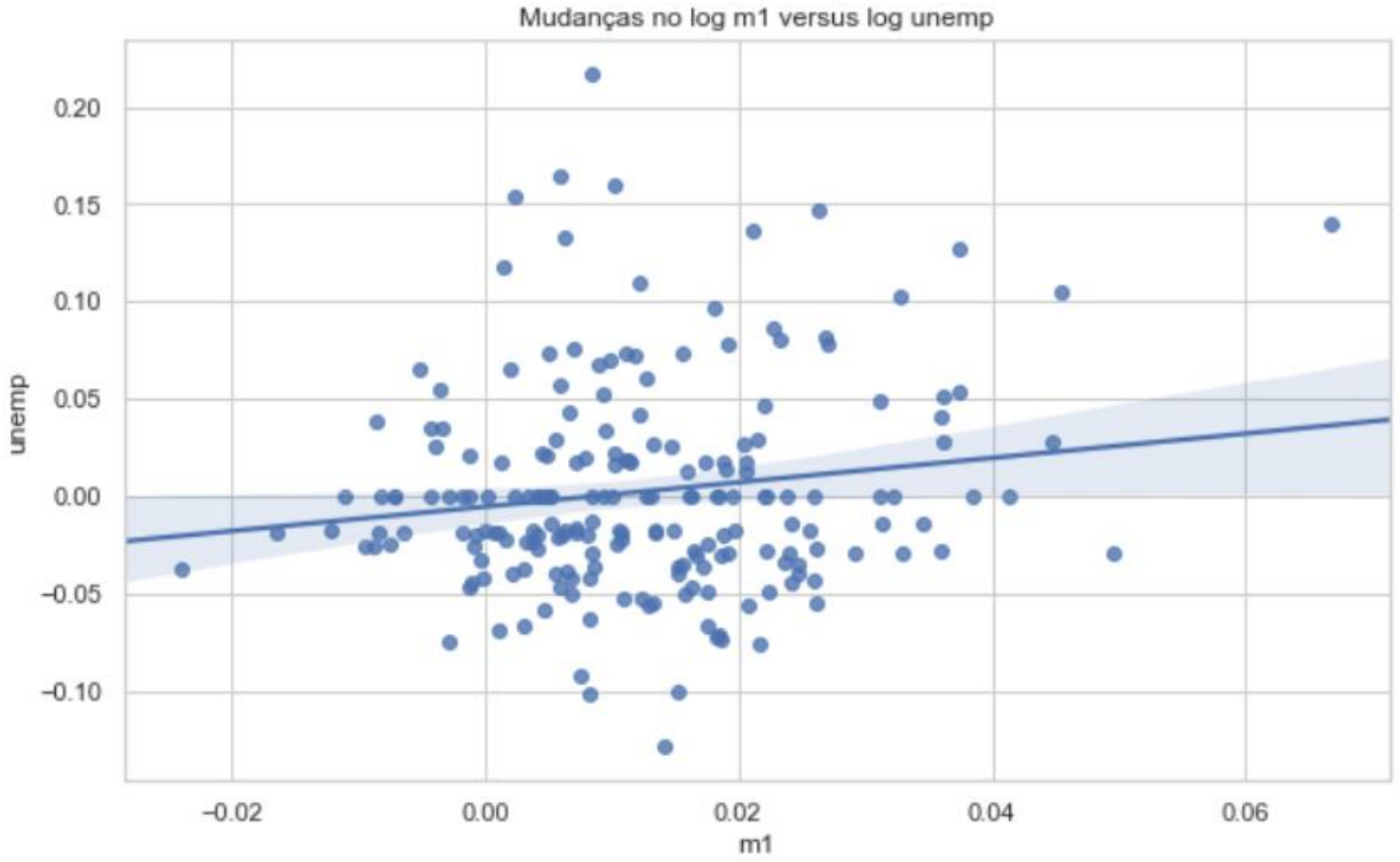
Histogramas e Plotagens de Densidade



- Plotagens de Dispersão ou de Pontos

  - Plotagem de pontos ou de dispersão podem ser uma maneira conveniente de analisar o relacionamento entre duas séries de dados unidimensionais;
  - A partir de um conjunto de dados utiliza-se o método regplot() da Seaborn, que gera uma plotagem de dispersão e inclui um traço de regressão linear.

Plotagens de Dispersão ou de Pontos

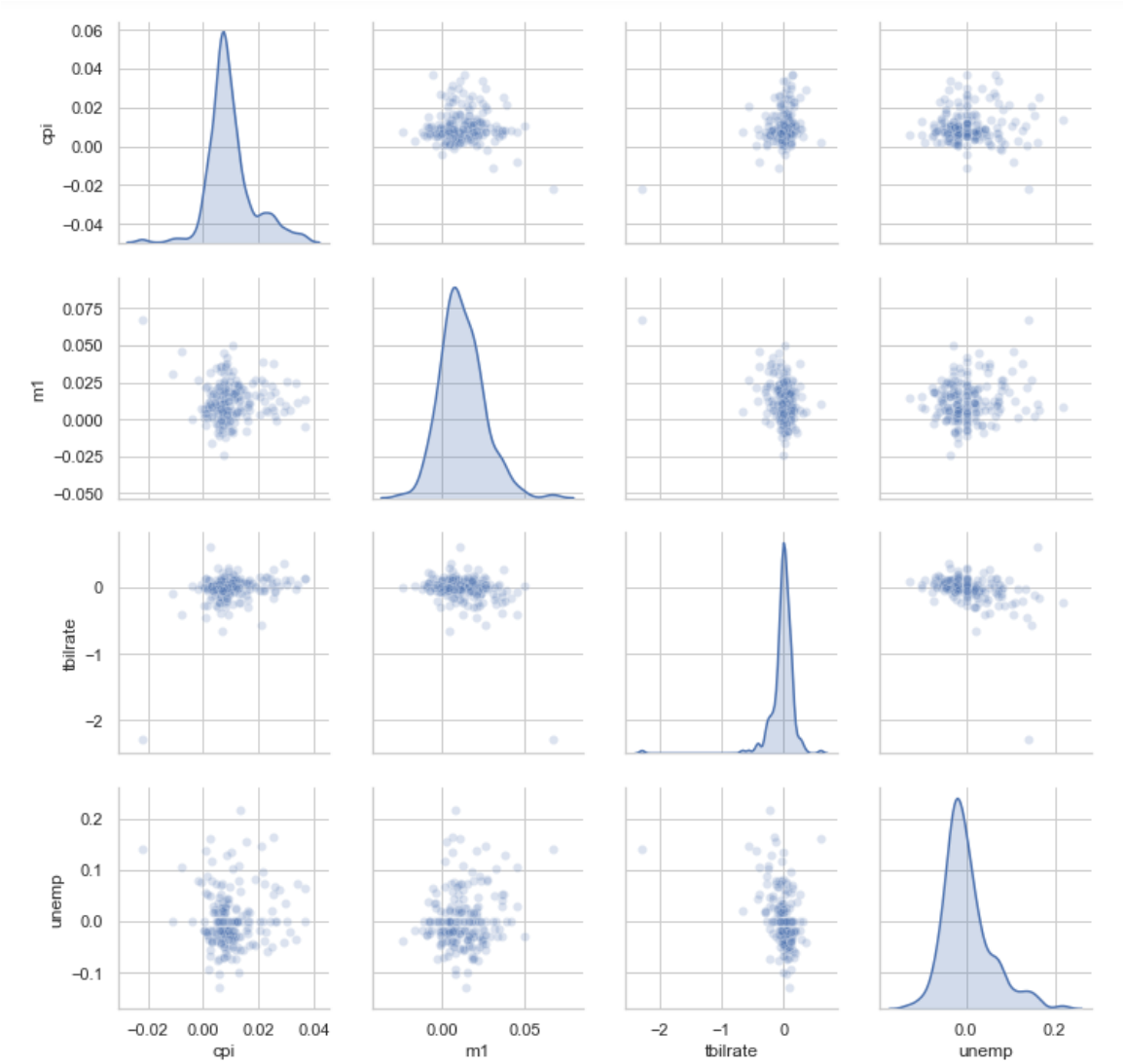


Plotagens de Dispersão ou de Pontos

- Na análise exploratória de dados é conveniente observar todas as plotagens de dispersão entre um grupo de variáveis;
- Isso é conhecido como plotagem de pares ou uma matriz de plotagem de dispersão.

```
sns.pairplot(t_data, diag_kind='kde')
```

Plotagens de Dispersão ou de Pontos

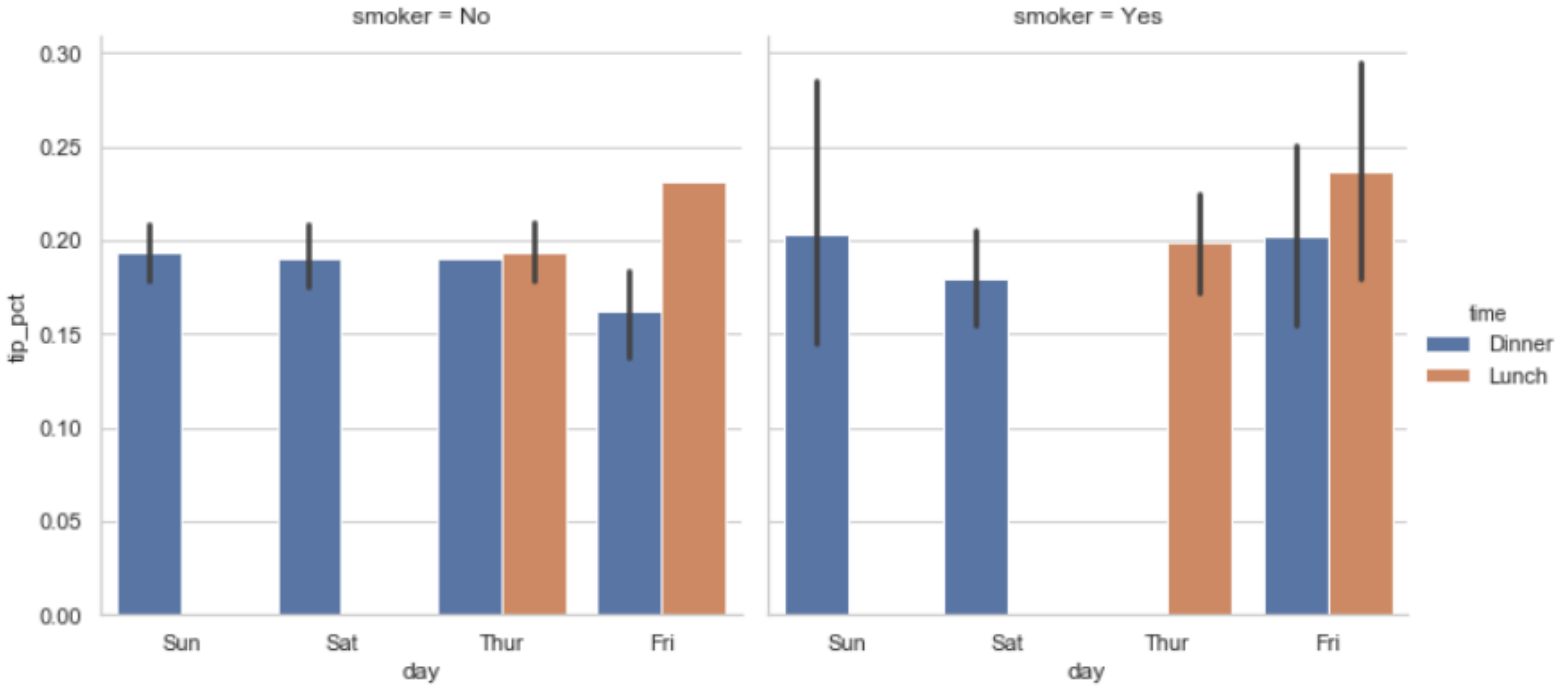


Grades de Faceta e Dados de Categoria

- Uma forma de visualizar dados com muitas variáveis de categoria é por meio de uma grade de faceta (*facet grid*).

```
sns.factorplot(x='day', y='tip_pct', hue='time', col='smoker',
               kind='bar', data=tips[tips.tip_pct > 0.2])
```

Grades de Faceta e Dados de Categoria

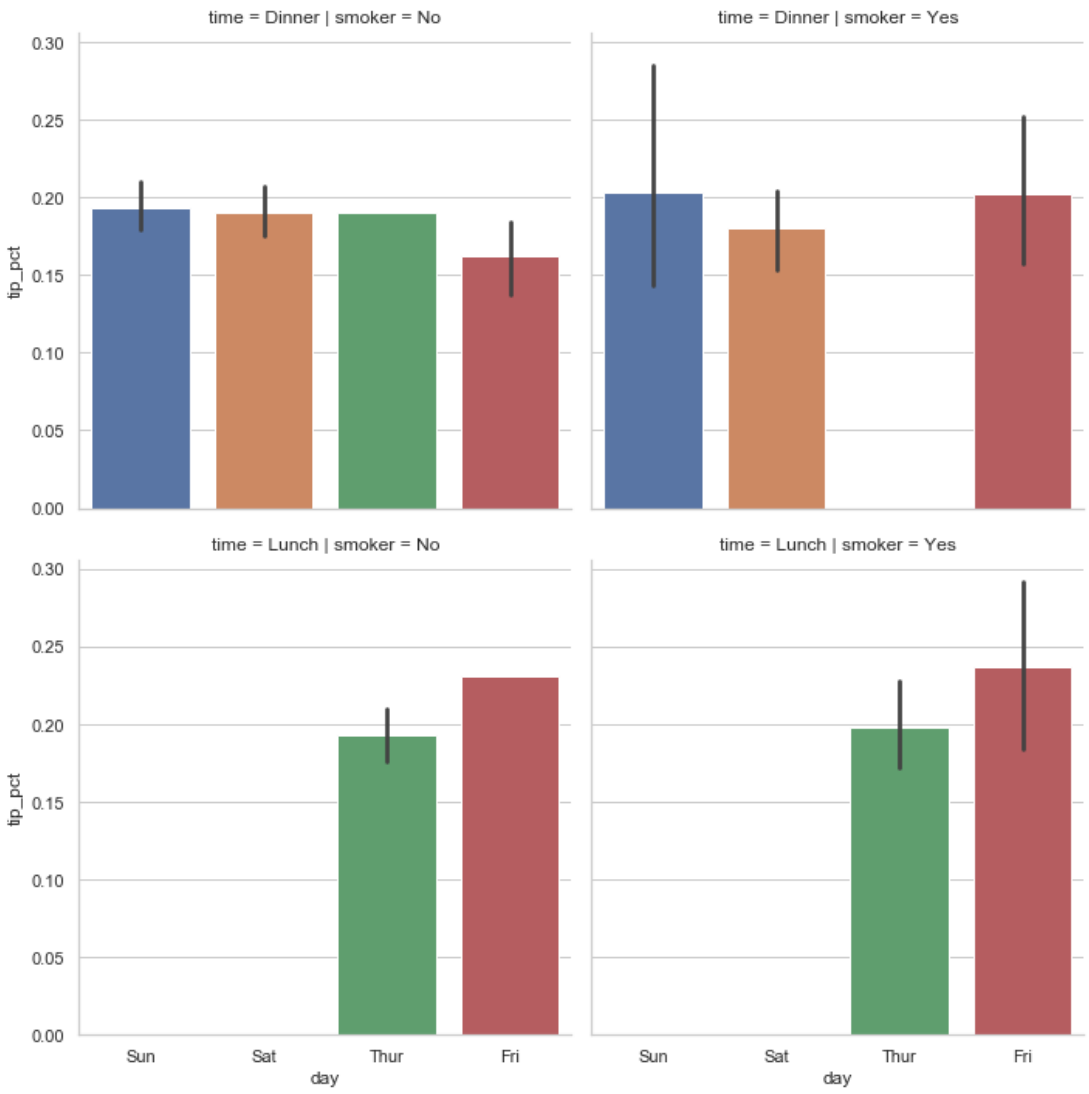


Grades de Faceta e Dados de Categoria

- Em vez de agrupar por ‘time’ e por cores diferentes de barra em uma faceta, pode-se também expandir a grade de facetas acrescentando uma linha por valor de ‘time’.

```
sns.factorplot(x='day', y='tip_pct', row='time',
               col='smoker',
               kind='bar', data=tips[tips.tip_pct < 1])
```

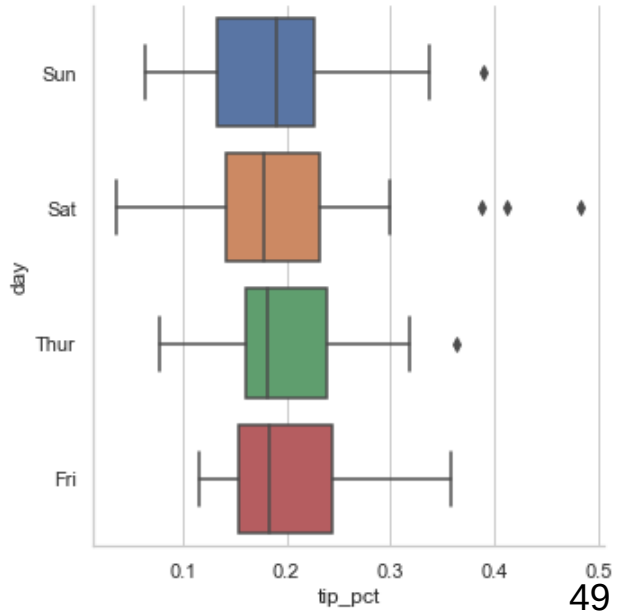
Grades de Faceta e Dados de Categoria



Grades de Faceta e Dados de Categoria

- factorplot permite outros tipos de plotagens que podem ser úteis, dependendo do que se deseja exibir.
- Um exemplo é a plotagem de caixa (box plot), que apresenta a mediana, os quartis e os valores discrepantes.

```
sns.factorplot(x='tip_pct', y='day', kind='box',
               data=tips[tips.tip_pct < 0.5])
```



Conclusão

- Em geral sistemas necessitam de plotagens e visualizações que podem ser simples ou complexas;
- Nesta apresentação foram demonstradas algumas visualizações básicas de dados usando as bibliotecas Pandas, matplotlib e Seaborn;
- Outras possibilidades são as bibliotecas Bokeh (bokeh.pydata.org) e Plotly (https://github.com/plotly/plotly.py).

Exemplo de Análise de Dados

- A Federal Election Commission (Comissão Eleitoral Federal) dos Estados Unidos publica dados sobre contribuições para campanhas públicas;
- Os dados incluem os nomes dos colaboradores, a profissão e o empregador, o endereço e o valor da contribuição;
- Para a eleição de 2012 foi disponibilizado um conjunto de dados com 150 megabytes chamado P00000001-ALL.csv.

- Exercício

  - Com base no exemplo anterior elabore uma análise a partir de um conjunto de dados que deve obrigatoriamente apresentar o resultado das análises na forma de gráficos.