Aula 7B - Visualização Gráfica de Dados

Gustavo Oliveira e Andréa Rocha

Departamento de Computação Científica / UFPB

Julho de 2020

1 Visualização Gráfica de Dados

1.1 As bibliotecas *matplotlib* e *plotly*

Vamos começar refazendo os gráficos que fizemos anteriormente com o método **plot** dos *DataFrames* e *Series* utilizando as funções do **matplotlib.pyplot**.

- O **matplotlib** transforma os dados em gráficos através de duas componentes: **figuras** (por exemplo janelas) e **eixos** (uma região onde os pontos podem ser determinados por meio de coordenadas). Se temos uma figura bidimensional, tipicamente os eixos são x-y, mas podemos ter coordenadas polares também. Se temos uma figura tridimensional, os eixos tipicamente são x-y-z, mas também podemos ter coordenadas esféricas, cilíndricas, etc.
- Como as figuras são determinadas pelas posições no plano ou no espaço, utilizamos com mais frequência os **eixos** de um objeto do **matplotlib**.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # Aqui utilizaremos a biblioteca matplotlib
```

[2]:
```
serie_Idade = pd.Series({'Ana':20, 'João': 19, 'Maria': 21, 'Pedro': 22,␣
↪'Túlio': 20}, name="Idade")
serie_Peso = pd.Series({'Ana':55, 'João': 80, 'Maria': 62, 'Pedro': 67, 'Túlio':
↪ 73}, name="Peso")
serie_Altura = pd.Series({'Ana':162, 'João': 178, 'Maria': 162, 'Pedro': 165,␣
↪'Túlio': 171}, name="Altura")
```

[3]:
```
dicionario_series_exemplo = {'Idade': serie_Idade, 'Peso': serie_Peso, 'Altura':
↪ serie_Altura}
```

[4]:
```
df_dict_series = pd.DataFrame(dicionario_series_exemplo);df_dict_series
```

[4]:

	Idade	Peso	Altura
Ana	20	55	162
João	19	80	178
Maria	21	62	162
Pedro	22	67	165

```
Túlio     20    73     171
```

[5]:
```
df_exemplo = pd.read_csv('exemplo_data.csv', index_col=0)
df_exemplo['coluna_3'] = pd.Series([1,2,3,4,5,6,7,8,np.nan,np.
 ↪nan],index=df_exemplo.index)
df_exemplo
```

[5]:
```
            coluna_1  coluna_2  coluna_3
2020-01-01 -0.416092  1.810364       1.0
2020-01-02 -0.137970  2.578520       2.0
2020-01-03  0.575827  0.060866       3.0
2020-01-04 -0.017367  1.299587       4.0
2020-01-05  1.384279 -0.381732       5.0
2020-01-06  0.549706 -1.308789       6.0
2020-01-07 -0.282296 -1.688979       7.0
2020-01-08 -0.989730 -0.028121       8.0
2020-01-09  0.275582 -0.177659       NaN
2020-01-10  0.685132  0.502535       NaN
```

[6]:
```
covid_PB = pd.read_csv('https://superset.plataformatarget.com.br/superset/
 ↪explore_json/?form_data=%7B%22slice_id%22%3A1550%7D&csv=true',
                            sep=';', index_col=0)
covid_PB.head()
```

[6]:
```
            casosAcumulados  casosNovos  descartados  recuperados  \
data
2020-07-21            68844        1164        78605        25028
2020-07-20            67680         298        76190        24486
2020-07-19            67382         411        76186        24439
2020-07-18            66971         624        76176        24437
2020-07-17            66347         924        76102        24390

            obitosAcumulados  obitosNovos  Letalidade
data
2020-07-21              1558           41    0.022631
2020-07-20              1517           31    0.022414
2020-07-19              1486            9    0.022053
2020-07-18              1477           31    0.022054
2020-07-17              1446           28    0.021795
```

[7]:
```
covid_BR = pd.read_csv("HIST_PAINEL_COVIDBR_18jul2020.csv", low_memory=False)
covid_BR.head()
```

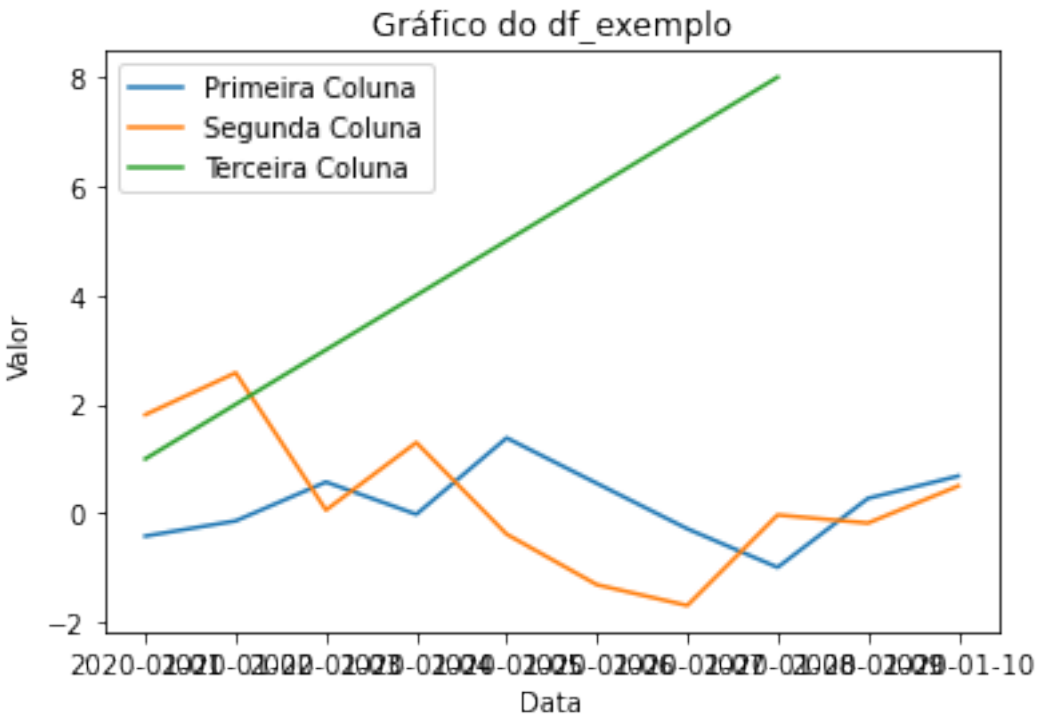
[7]:
```
   Unnamed: 0  regiao estado municipio  coduf  codmun  codRegiaoSaude  \
0           0  Brasil    NaN       NaN     76     NaN             NaN
1           1  Brasil    NaN       NaN     76     NaN             NaN
2           2  Brasil    NaN       NaN     76     NaN             NaN
```

```
3           3  Brasil    NaN       NaN     76     NaN             NaN
4           4  Brasil    NaN       NaN     76     NaN             NaN

  nomeRegiaoSaude        data  semanaEpi populacaoTCU2019  casosAcumulado  \
0             NaN  2020-02-25          9        210147125               0
1             NaN  2020-02-26          9        210147125               1
2             NaN  2020-02-27          9        210147125               1
3             NaN  2020-02-28          9        210147125               1
4             NaN  2020-02-29          9        210147125               2

   casosNovos  obitosAcumulado  obitosNovos  Recuperadosnovos  \
0           0                0            0               NaN
1           1                0            0               NaN
2           0                0            0               NaN
3           0                0            0               NaN
4           1                0            0               NaN

   emAcompanhamentoNovos  interior/metropolitana
0                    NaN                     NaN
1                    NaN                     NaN
2                    NaN                     NaN
3                    NaN                     NaN
4                    NaN                     NaN
```

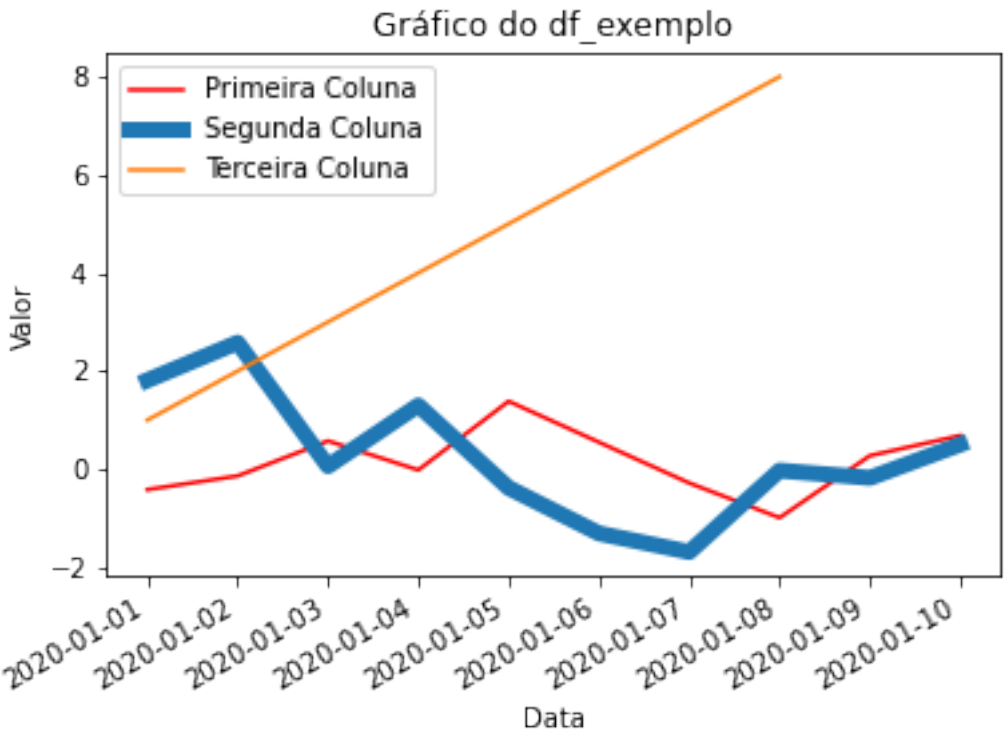
1.2 Gráfico de Linhas

[8]:
```
fig, ax = plt.subplots()  # Este comando cria uma figura com um eixo
ax.plot(df_exemplo.index, df_exemplo['coluna_1'], label = 'Primeira Coluna') #␣
 ↪Inserimos a linha relativa à coluna 1
ax.plot(df_exemplo.index, df_exemplo['coluna_2'], label = 'Segunda Coluna') #␣
 ↪Inserimos a linha relativa à coluna 2
ax.plot(df_exemplo.index, df_exemplo['coluna_3'], label = 'Terceira Coluna') #␣
 ↪Inserimos a linha relativa à coluna 3
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Valor') # Rótulo do eixo y
ax.set_title("Gráfico do df_exemplo")
ax.legend()
```

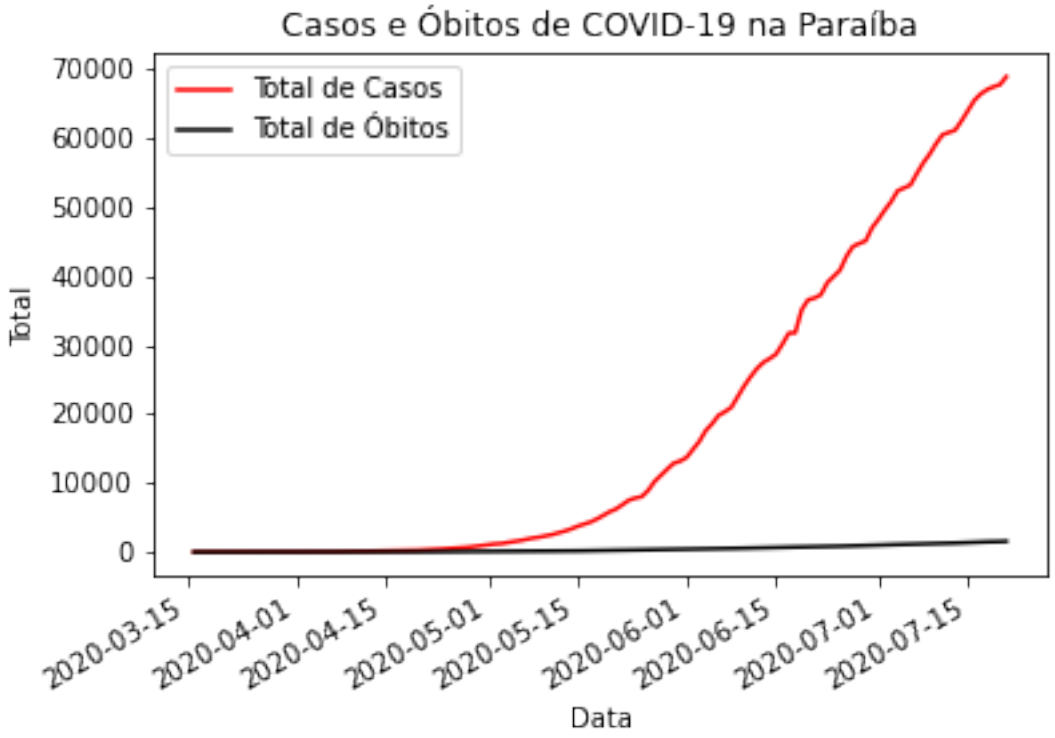
[8]:
```
<matplotlib.legend.Legend at 0x2080d6e0d48>
```



[9]:
```
fig, ax = plt.subplots()  # Este comando cria uma figura com um eixo
ax.plot(df_exemplo.index, df_exemplo['coluna_1'], label = 'Primeira Coluna',
        color = 'red') # Inserimos a linha relativa à coluna 1, definimos a cor␣
 ↪vermelha
ax.plot(df_exemplo.index, df_exemplo['coluna_2'],
        label = 'Segunda Coluna', linewidth=6.0) # Inserimos a linha relativa à␣
 ↪coluna 2 e aumentamos a grossura da linha
ax.plot(df_exemplo.index, df_exemplo['coluna_3'], label = 'Terceira Coluna') #␣
 ↪Inserimos a linha relativa à coluna 3
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Valor') # Rótulo do eixo y
ax.set_title("Gráfico do df_exemplo")
ax.legend()
fig.autofmt_xdate()
```



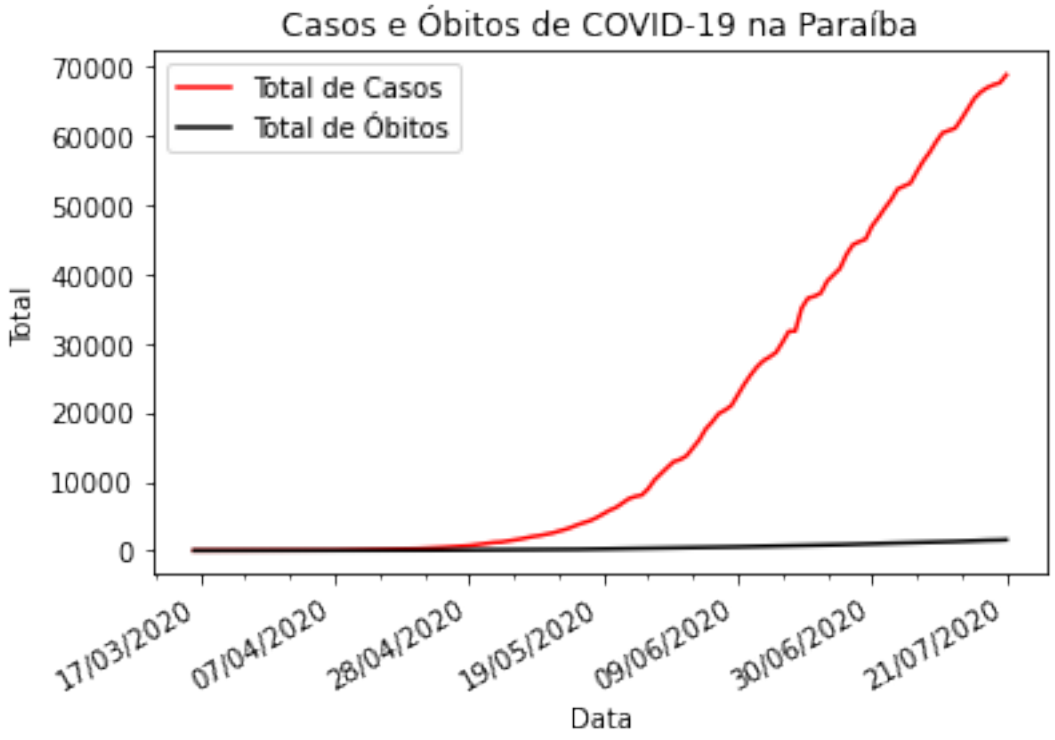
[10]:
```
covid_PB.index = pd.to_datetime(covid_PB.index)
covid_PB_casos_obitos = covid_PB[['casosAcumulados', 'obitosAcumulados']].
 ↪sort_index()
fig, ax = plt.subplots()
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['casosAcumulados'],␣
 ↪label = 'Total de Casos',
        color = 'red')
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['obitosAcumulados'],
        label = 'Total de Óbitos', color = 'black')
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Total') # Rótulo do eixo y
ax.set_title("Casos e Óbitos de COVID-19 na Paraíba")
ax.legend()
fig.autofmt_xdate()
```



Podemos alterar a apresentação das datas utilizando o subpacote *dates* do *matplotlib*.

[11]:
```
import matplotlib.dates as mdates
```

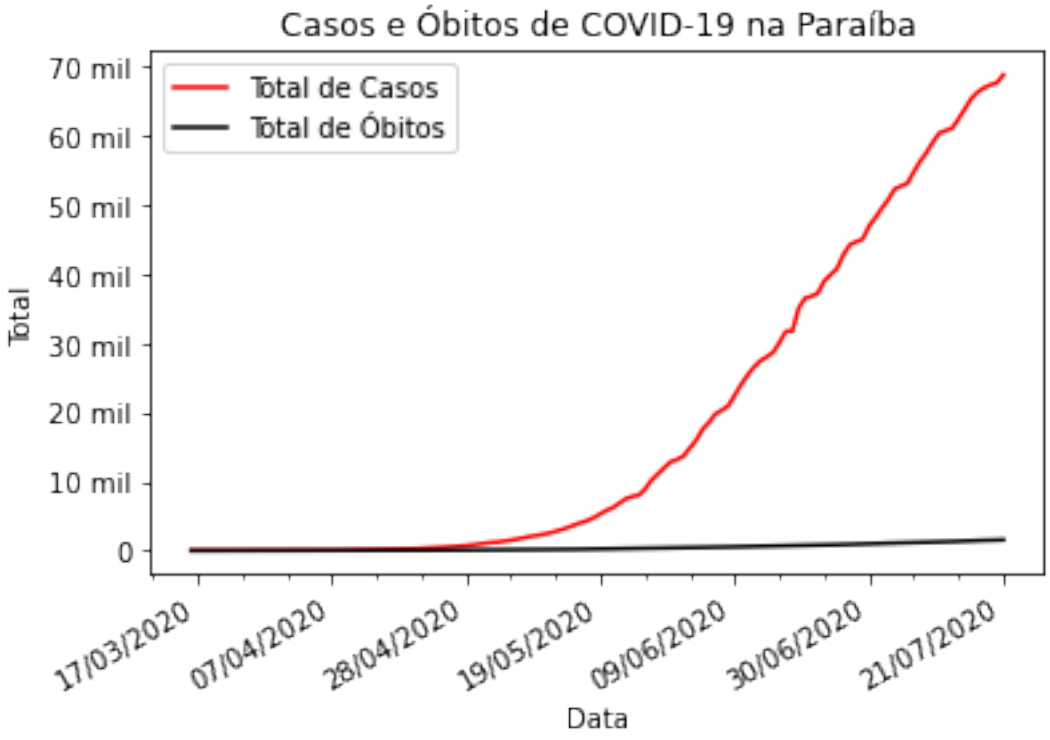
[12]:
```
fig, ax = plt.subplots()
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['casosAcumulados'],␣
 ↪label = 'Total de Casos', color = 'red')
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['obitosAcumulados'],␣
 ↪label = 'Total de Óbitos', color = 'black')
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Total') # Rótulo do eixo y
ax.set_title("Casos e Óbitos de COVID-19 na Paraíba")
ax.legend()
ax.xaxis.set_minor_locator(mdates.DayLocator(interval=7)) #Intervalo entre os␣
 ↪tracinhos
ax.xaxis.set_major_locator(mdates.DayLocator(interval=21)) #Intervalo entre as␣
 ↪datas
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%Y')) #Formato da data
fig.autofmt_xdate()
```



Vamos agora alterar o formato dos números do eixo vertical. Para tanto iremos definir uma função para realizar a formatação e utilizaremos a função *FuncFormatter* do subpacote *matplotlib.ticker*.

[13]:
```
from matplotlib.ticker import FuncFormatter
```

[14]:
```
def inserir_mil(x, pos):
    return ('{:.0f} mil'.format(x*1e-3)) if x!= 0 else 0
fig, ax = plt.subplots()
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['casosAcumulados'],␣
 ↪label = 'Total de Casos', color = 'red')
ax.plot(covid_PB_casos_obitos.index, covid_PB_casos_obitos['obitosAcumulados'],␣
 ↪label = 'Total de Óbitos', color = 'black')
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Total') # Rótulo do eixo y
ax.set_title("Casos e Óbitos de COVID-19 na Paraíba")
ax.legend()
ax.xaxis.set_minor_locator(mdates.DayLocator(interval=7)) #Intervalo entre os␣
 ↪tracinhos
ax.xaxis.set_major_locator(mdates.DayLocator(interval=21)) #Intervalo entre as␣
 ↪datas
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%Y')) #Formato da data
fig.autofmt_xdate()
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
```



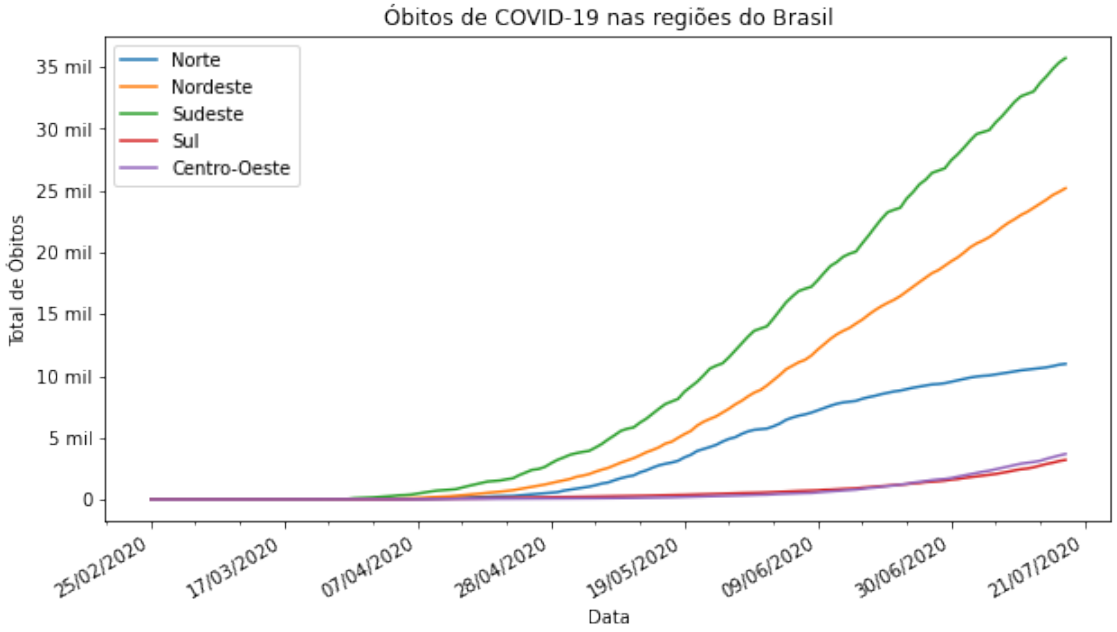
[15]:
```
covid_regioes = pd.DataFrame()

regioes = covid_BR.query('regiao != "Brasil"')['regiao'].drop_duplicates().
 ↪array;regioes

for regiao in regioes:
    temp_series = covid_BR.set_index('data').query('regiao ==␣
 ↪@regiao')['obitosAcumulado'].groupby('data').sum()/2
    #Obs.: Utilizamos @ na frente do nome da variável para utilizar o valor da␣
 ↪variável no query.
    #Obs.: Dividimos por 2, pois os óbitos estão sendo contados duas vezes,
    #uma para quando codmun == nan e outra quando não é nulo
    temp_series.name = 'obitos_' + regiao
    covid_regioes = pd.concat([covid_regioes, temp_series], axis=1)

covid_regioes.index = pd.to_datetime(covid_regioes.index)
covid_regioes.tail()
```

[15]:

data	obitos_Norte	obitos_Nordeste	obitos_Sudeste	obitos_Sul	\
2020-07-14	10628.0	23925.0	33718.0	2740.0	
2020-07-15	10693.0	24272.0	34246.0	2870.0	
2020-07-16	10790.0	24645.0	34857.0	2975.0	

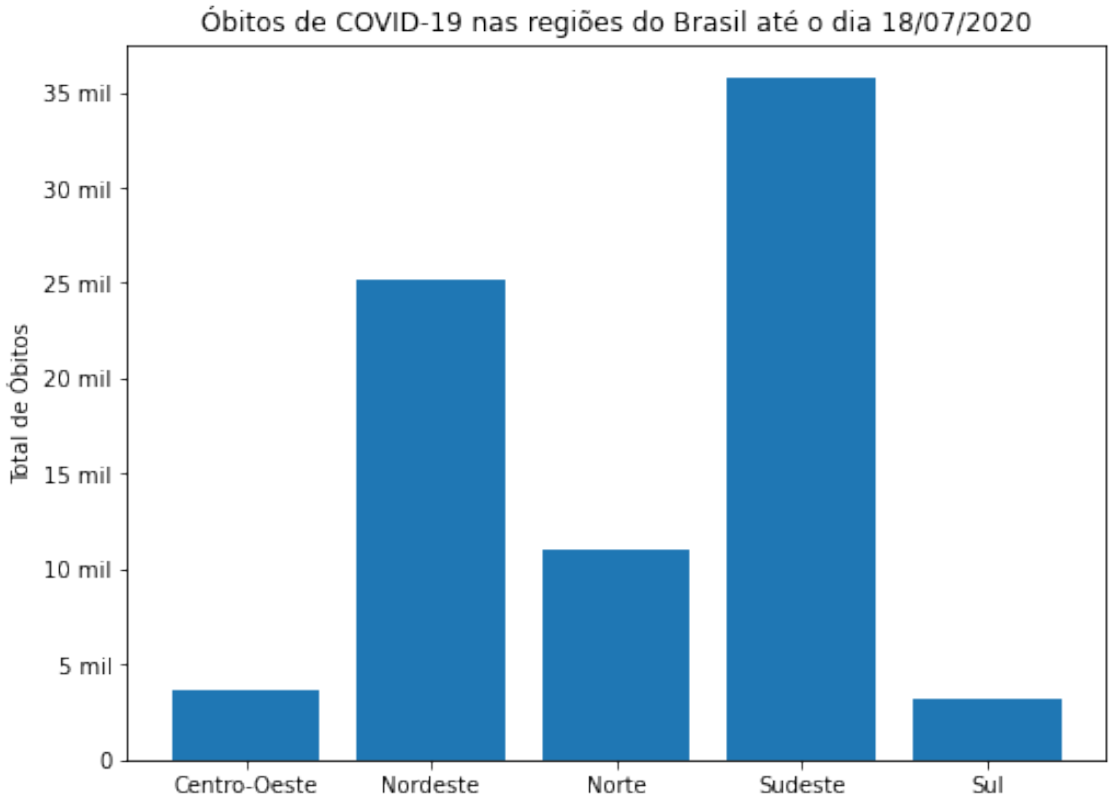
```
2020-07-17       10911.0          24902.0          35374.0       3104.0
2020-07-18       10972.0          25194.0          35732.0       3199.0

            obitos_Centro-Oeste
data
2020-07-14               3122.0
2020-07-15               3285.0
2020-07-16               3421.0
2020-07-17               3560.0
2020-07-18               3675.0
```

[16]:
```
fig, ax = plt.subplots(figsize=(10,5.5))
ax.plot(covid_regioes.index, covid_regioes['obitos_Norte'], label = 'Norte')
ax.plot(covid_regioes.index, covid_regioes['obitos_Nordeste'], label =␣
↪'Nordeste')
ax.plot(covid_regioes.index, covid_regioes['obitos_Sudeste'], label = 'Sudeste')
ax.plot(covid_regioes.index, covid_regioes['obitos_Sul'], label = 'Sul')
ax.plot(covid_regioes.index, covid_regioes['obitos_Centro-Oeste'], label =␣
↪'Centro-Oeste')
ax.set_xlabel('Data') # Rótulo do eixo x
ax.set_ylabel('Total de Óbitos') # Rótulo do eixo y
ax.set_title("Óbitos de COVID-19 nas regiões do Brasil")
ax.legend()
ax.xaxis.set_minor_locator(mdates.DayLocator(interval=7)) #Intervalo entre os␣
↪tracinhos
ax.xaxis.set_major_locator(mdates.DayLocator(interval=21)) #Intervalo entre as␣
↪datas
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%Y')) #Formato da data
fig.autofmt_xdate()
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
```



1.3 Gráfico de Colunas e de Linhas

[17]:
```
covid_Regioes = covid_BR[['regiao','obitosNovos']].groupby('regiao').sum().
 ↪query('regiao != "Brasil"')/2
fig, ax = plt.subplots(figsize=(8,6))
ax.bar(covid_Regioes.index, covid_Regioes['obitosNovos'])
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
ax.set_ylabel('Total de Óbitos') # Rótulo do eixo y
_ = ax.set_title("Óbitos de COVID-19 nas regiões do Brasil até o dia 18/07/
 ↪2020")
```

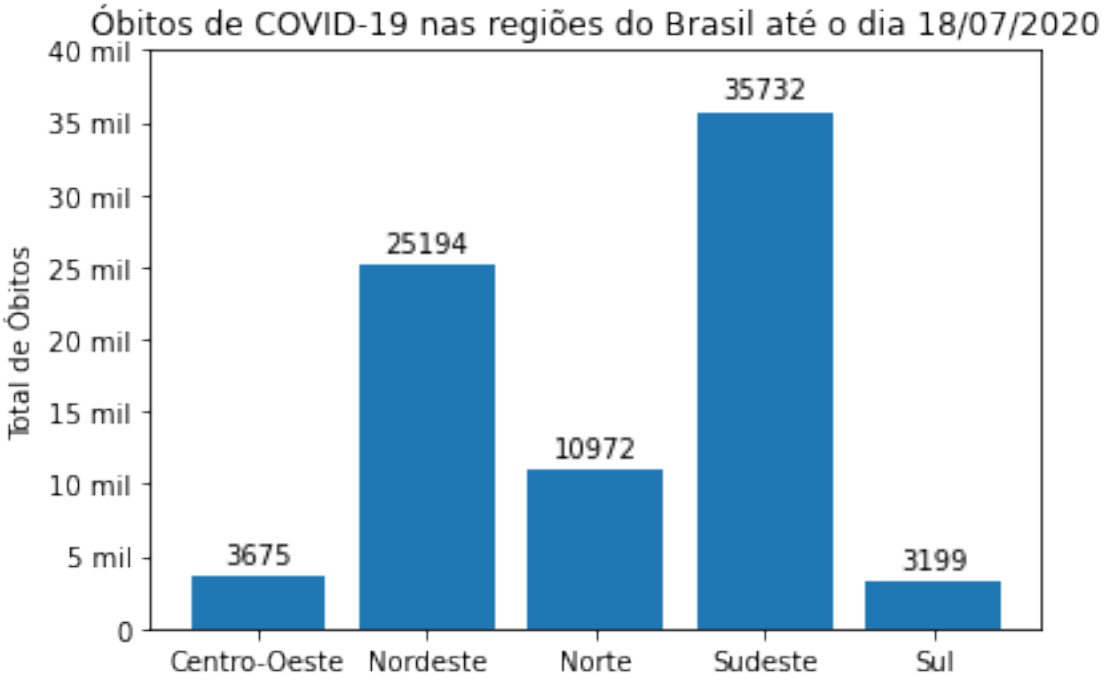


- Podemos inserir o total de cada região em cima da coluna correspondente. Para tanto, utilizaremos a seguinte função *autolabel* disponível na página do *matplotlib*:

[18]:
```
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        #ax.annotate('{}'.format(height), #antigo
        ax.annotate('{:.0f}'.format(height), #Modificamos para apresentar o␣
 ↪número inteiro
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
```

[19]:
```
covid_Regioes = covid_BR[['regiao','obitosNovos']].groupby('regiao').sum().
 ↪query('regiao != "Brasil"')/2
fig, ax = plt.subplots()
plt.ylim(0, 40000) # aumentamos o limite da coordenada y
retangulos = ax.bar(covid_Regioes.index, covid_Regioes['obitosNovos'])
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
```

```
ax.set_ylabel('Total de Óbitos') # Rótulo do eixo y
ax.set_title("Óbitos de COVID-19 nas regiões do Brasil até o dia 18/07/2020")
autolabel(retangulos)
```



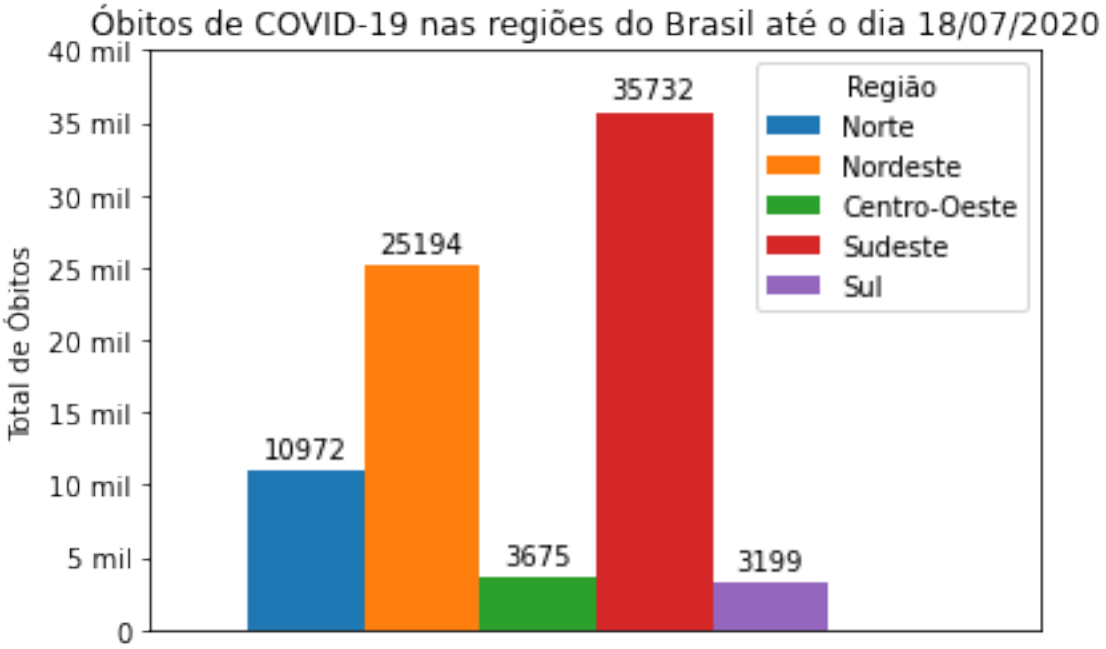
- Para realizarmos os "plots" agrupados das barras devemos realizar 5 "plots" distintos, um para cada barra.
- Cada plot sofrerá uma translação (exceto o do meio).
- Iremos reduzir a largura de cada barra.

[20]:
```
covid_Regioes = covid_BR[['regiao','obitosNovos']].groupby('regiao').sum().
 ↪query('regiao != "Brasil"')/2
largura = 0.3
```

[21]:
```
fig, ax = plt.subplots()
retangulo1 = ax.bar([-2*largura], covid_Regioes.loc[['Norte'],['obitosNovos']].
 ↪to_numpy()[0], largura, label='Norte')
retangulo2 = ax.bar([-largura], covid_Regioes.loc[['Nordeste'],['obitosNovos']].
 ↪to_numpy()[0], largura, label='Nordeste')
retangulo3 = ax.bar([0], covid_Regioes.loc[['Centro-Oeste'],['obitosNovos']].
 ↪to_numpy()[0], largura, label='Centro-Oeste')
retangulo4 = ax.bar([largura], covid_Regioes.loc[['Sudeste'],['obitosNovos']].
 ↪to_numpy()[0], largura, label='Sudeste')
```

```
retangulo5 = ax.bar([2*largura], covid_Regioes.loc[['Sul'],['obitosNovos']].
 ↪to_numpy()[0], largura, label='Sul')
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
ax.set_ylabel('Total de Óbitos') # Rótulo do eixo y
ax.set_title("Óbitos de COVID-19 nas regiões do Brasil até o dia 18/07/2020")
autolabel(retangulo1); autolabel(retangulo2); autolabel(retangulo3);␣
 ↪autolabel(retangulo4); autolabel(retangulo5)
plt.ylim(0, 40000) # Aumentamos o limite da coordenada y
plt.xlim(-1, 1.3) # Modificamos os limites da coordenada x
plt.xticks([], []) # Remover os "ticks" no eixo x
#plt.xticks([0], ['Região']) # Se quisermos incluir o rótulo "Região" na␣
 ↪posição 0 do eixo x
ax.legend(title="Região")
```

[21]: <matplotlib.legend.Legend at 0x2080f2f6848>

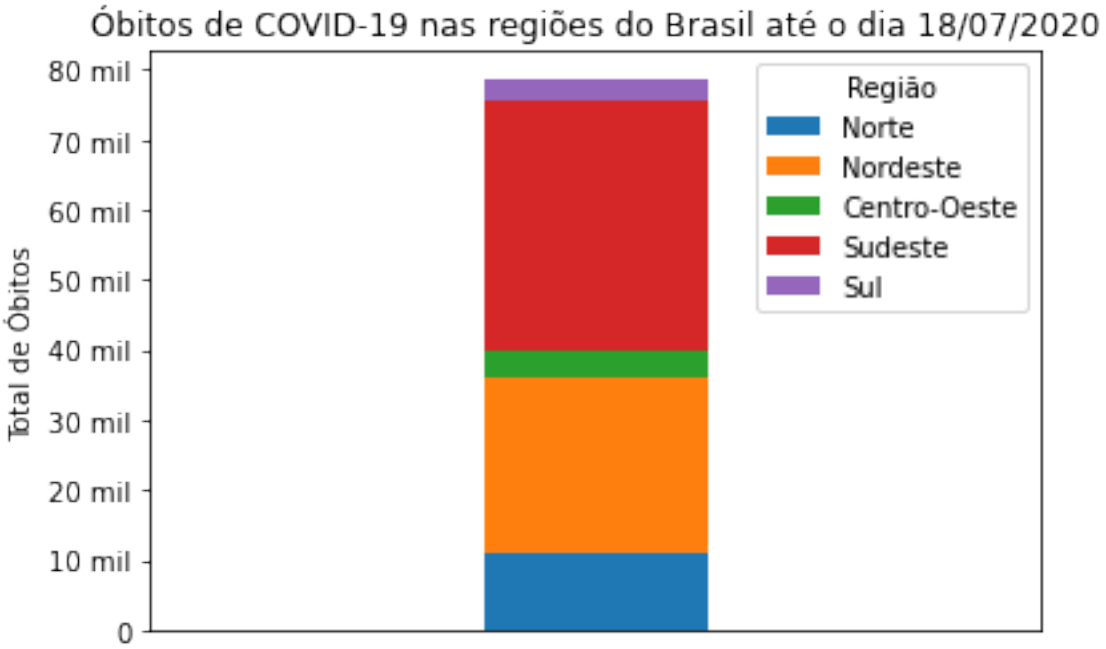


- Para empilharmos as barras manualmente devemos utilizar o argumento **bottom**:

[22]:
```
largura = 0.25
obitos_norte = covid_Regioes.loc[['Norte'],['obitosNovos']].to_numpy()[0]
obitos_nordeste = covid_Regioes.loc[['Nordeste'],['obitosNovos']].to_numpy()[0]
obitos_centro_oeste = covid_Regioes.loc[['Centro-Oeste'],['obitosNovos']].
 ↪to_numpy()[0]
obitos_sudeste = covid_Regioes.loc[['Sudeste'],['obitosNovos']].to_numpy()[0]
obitos_sul = covid_Regioes.loc[['Sul'],['obitosNovos']].to_numpy()[0]
```

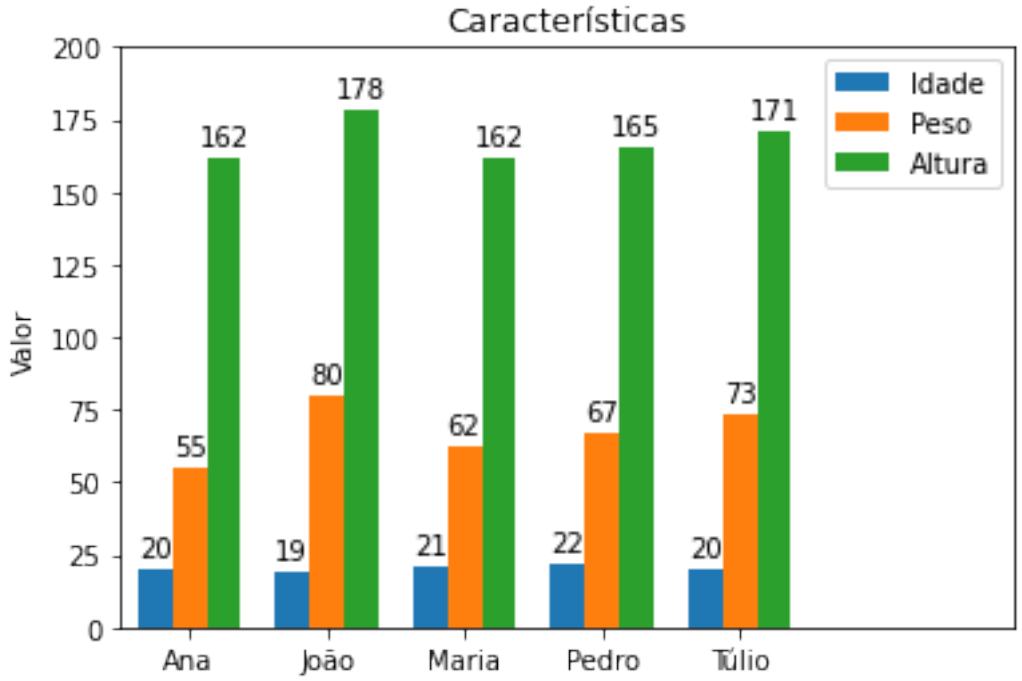
[23]:
```
fig, ax = plt.subplots()
retangulo1 = ax.bar([0.5], obitos_norte, largura, label='Norte')
retangulo2 = ax.bar([0.5], obitos_nordeste, largura, label='Nordeste', bottom =␣
 ↪obitos_norte)
retangulo3 = ax.bar([0.5], obitos_centro_oeste, largura, label='Centro-Oeste',␣
 ↪bottom = obitos_norte + obitos_nordeste)
retangulo4 = ax.bar([0.5], obitos_sudeste, largura, label='Sudeste', bottom =␣
 ↪obitos_norte +
                    obitos_nordeste + obitos_centro_oeste)
retangulo5 = ax.bar([0.5], obitos_sul, largura, label='Sul', bottom =␣
 ↪obitos_norte +
                    obitos_nordeste + obitos_centro_oeste + obitos_sudeste)
ax.yaxis.set_major_formatter(FuncFormatter(inserir_mil))
ax.set_ylabel('Total de Óbitos') # Rótulo do eixo y
ax.set_title("Óbitos de COVID-19 nas regiões do Brasil até o dia 18/07/2020")
plt.xticks([], [])
#plt.xticks([0], ['Região']) # Se quisermos incluir o rótulo "Região" na␣
 ↪posição 0 do eixo x
plt.xlim(0,1)
ax.legend(title="Região")
```

[23]:
```
<matplotlib.legend.Legend at 0x2080f38fb08>
```



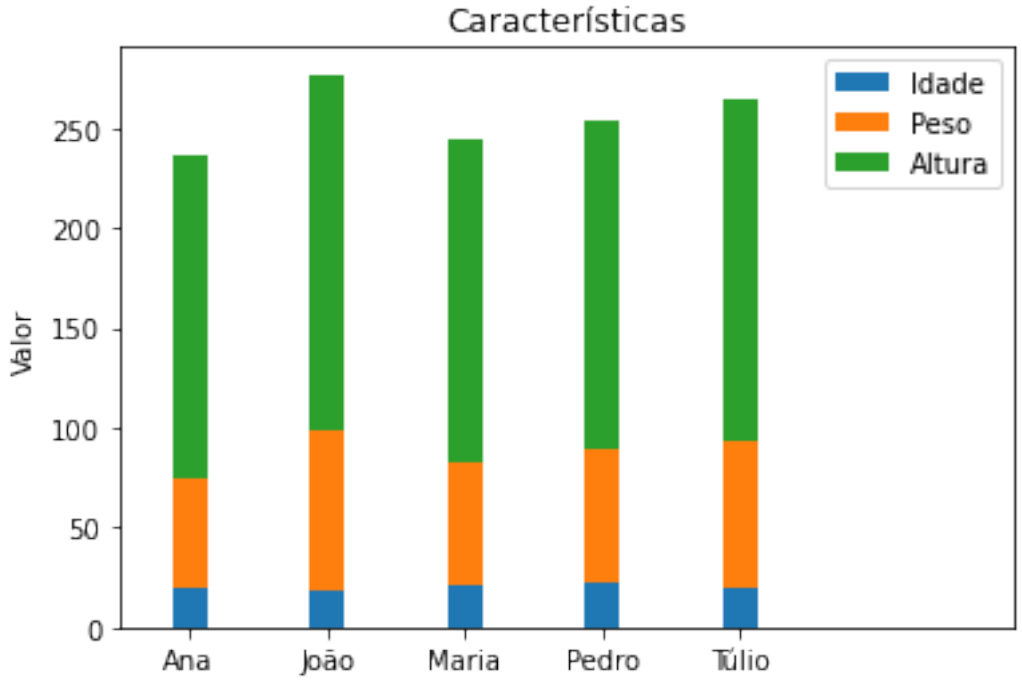
[24]:
```
x = np.arange(len(df_dict_series.index))
largura = 0.25
```

```
fig, ax = plt.subplots()
retangulo1 = ax.bar(x - largura, df_dict_series.Idade, largura, label='Idade')
retangulo2 = ax.bar(x, df_dict_series.Peso, largura, label='Peso')
retangulo3 = ax.bar(x + largura, df_dict_series.Altura, largura, label='Altura')
autolabel(retangulo1); autolabel(retangulo2); autolabel(retangulo3)
plt.ylim(0,200)
plt.xlim(-0.5,6)
ax.set_ylabel('Valor')
ax.set_title('Características')
ax.set_xticks(x)
ax.set_xticklabels(df_dict_series.index)
_ = ax.legend()
```



[25]:
```
x = np.arange(len(df_dict_series.index))
largura = 0.25
fig, ax = plt.subplots()
retangulo1 = ax.bar(x, df_dict_series.Idade, largura, label='Idade')
retangulo2 = ax.bar(x, df_dict_series.Peso, largura, label='Peso', bottom =␣
↪df_dict_series.Idade)
retangulo3 = ax.bar(x, df_dict_series.Altura, largura, label='Altura', bottom =␣
↪df_dict_series.Idade + df_dict_series.Peso)
plt.xlim(-0.5,6)
ax.set_ylabel('Valor')
ax.set_title('Características')
```

```
ax.set_xticks(x)
ax.set_xticklabels(df_dict_series.index)
_ = ax.legend()
```



1.4 Gráfico de Barras

- Para construir os gráficos de barras procedemos de maneira análoga ao que foi feito acima.
- Substituímos o método **bar** por **barh**
- Caso haja interesse deve modificar a função autolabel, alterando a altura, *height*, pela largura, *width*.

1.5 Gráfico de Setores

Neste caso devemos modificar o *DataFrame* para conter percentuais (ou pesos).

- Podemos usar os parâmetros:
- **autopct** que adiciona o percentual de cada "fatia".
- **shadow** que adiciona sombra
- **explode** que separa fatias selecionadas

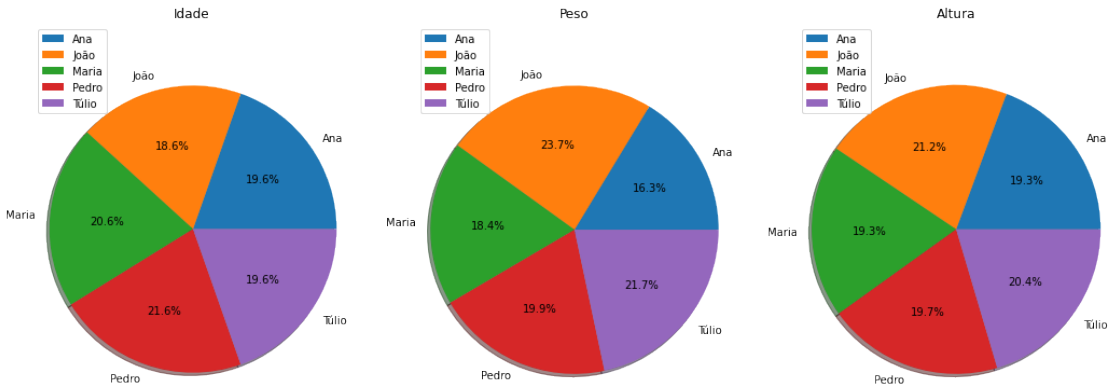
[26]:
```
df_dict_series_pct = df_dict_series.copy()
df_dict_series_pct.Idade = df_dict_series_pct.Idade/df_dict_series_pct.Idade.
↪sum()
```

```
df_dict_series_pct.Peso = df_dict_series_pct.Peso/df_dict_series_pct.Peso.sum()
df_dict_series_pct.Altura = df_dict_series_pct.Altura/df_dict_series_pct.Altura.
 ↪sum()
df_dict_series_pct
```

[26]:

	Idade	Peso	Altura
Ana	0.196078	0.163205	0.193317
João	0.186275	0.237389	0.212411
Maria	0.205882	0.183976	0.193317
Pedro	0.215686	0.198813	0.196897
Túlio	0.196078	0.216617	0.204057

[27]:
```
figs, axs = plt.subplots(1,3, figsize=(18,7)) #1 linha3 e 3 colunas de "plots"
axs[0].pie(df_dict_series_pct.Idade,  labels=df_dict_series_pct.index,␣
 ↪autopct='%1.1f%%', shadow=True)
axs[0].axis('equal')  # Igualando os eixos para garantir que obteremos um␣
 ↪círculo
axs[0].legend(loc = 'upper left')
axs[0].set_title('Idade')
axs[1].pie(df_dict_series_pct.Peso,  labels=df_dict_series_pct.index,␣
 ↪autopct='%1.1f%%', shadow=True)
axs[1].axis('equal')
axs[1].legend(loc = 'upper left')
axs[1].set_title('Peso')
axs[2].pie(df_dict_series_pct.Altura,  labels=df_dict_series_pct.index,␣
 ↪autopct='%1.1f%%', shadow=True)
axs[2].axis('equal')
axs[2].legend(loc = 'upper left')
_ = axs[2].set_title('Altura') #Atribuímos a uma variável para não termos saída
```

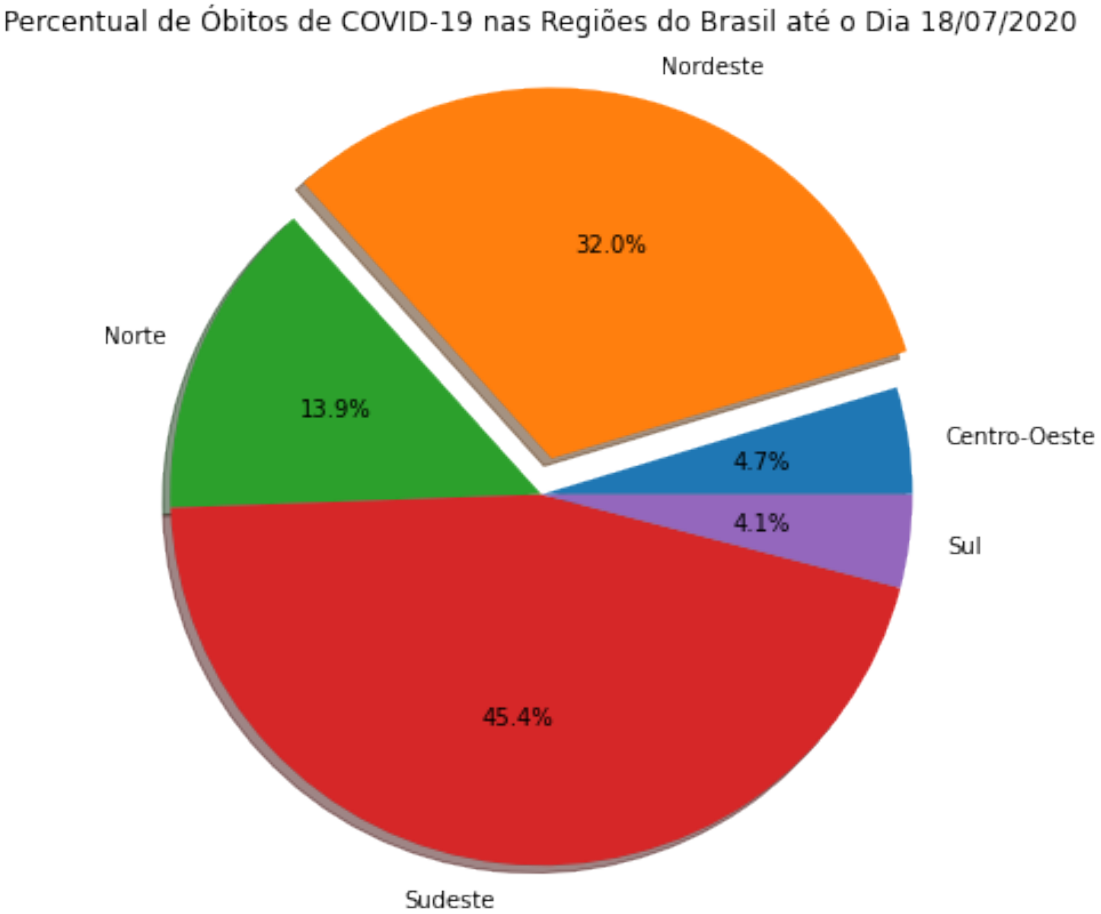


[28]:
```
covid_Regioes_pct = covid_Regioes/covid_Regioes.sum()
covid_Regioes_pct['explodir'] = covid_Regioes_pct.index.map(lambda regiao: 0.1␣
 ↪if regiao == 'Nordeste' else 0)
covid_Regioes_pct
```

[28]:

regiao	obitosNovos	explodir
Centro-Oeste	0.046654	0.0
Nordeste	0.319834	0.1
Norte	0.139288	0.0
Sudeste	0.453613	0.0
Sul	0.040611	0.0

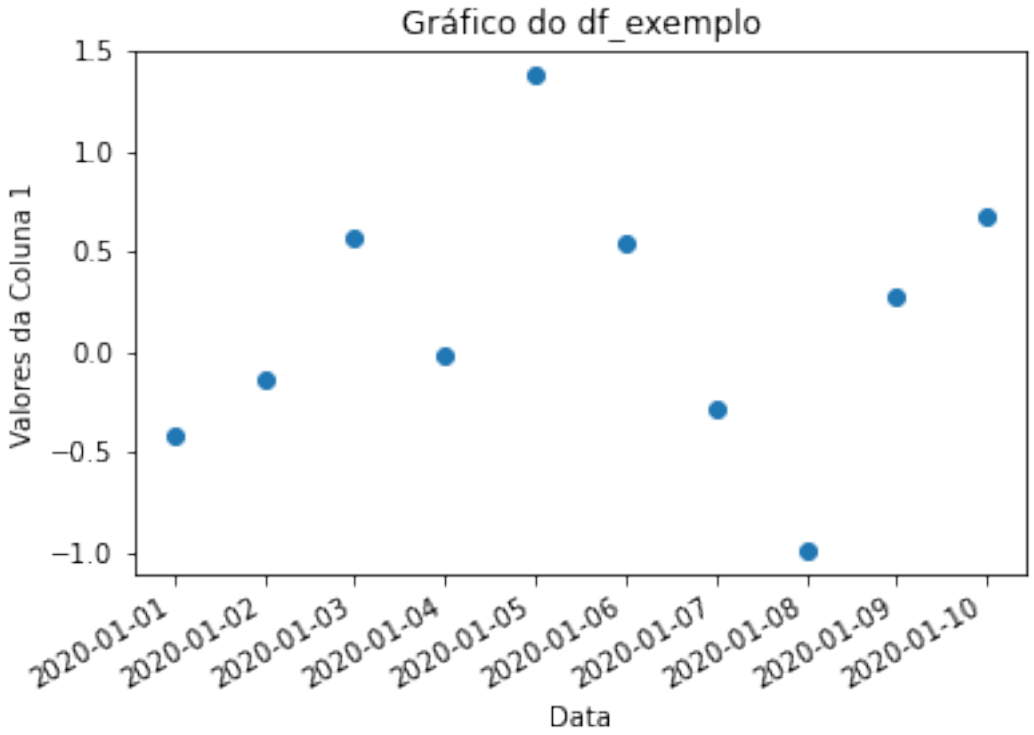
[29]:
```
fig, ax = plt.subplots(figsize = (7,7))
ax.pie(covid_Regioes_pct.obitosNovos, explode=covid_Regioes_pct.explodir,
       labels=covid_Regioes_pct.index, autopct='%1.1f%%', shadow=True)
ax.set_title('Percentual de Óbitos de COVID-19 nas Regiões do Brasil até o Dia␣
 ↪18/07/2020')
_ = ax.axis('equal')
```



1.6 Gráfico de Dispersão

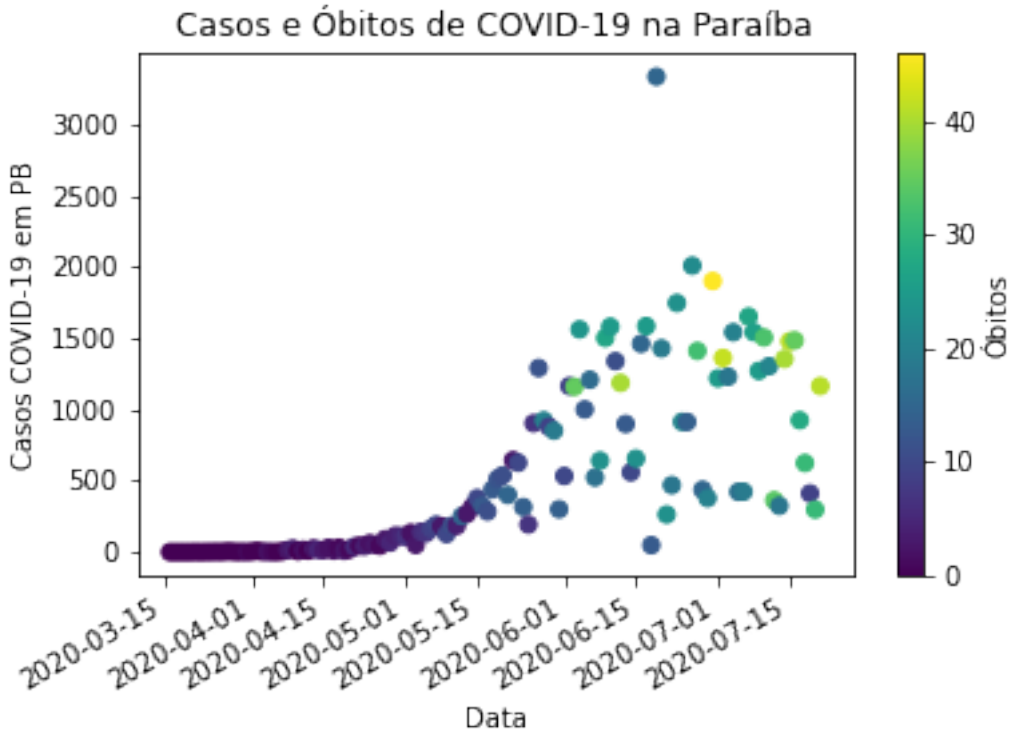
Para gráficos de dispersão vários argumentos são os mesmos que já vimos no método ***plot*** *do* pandas*.

[30]:
```
fig, ax = plt.subplots()
ax.scatter(df_exemplo.index, df_exemplo['coluna_1'])
fig.autofmt_xdate()
ax.set_xlabel('Data')
ax.set_ylabel('Valores da Coluna 1')
_= ax.set_title('Gráfico do df_exemplo')
```

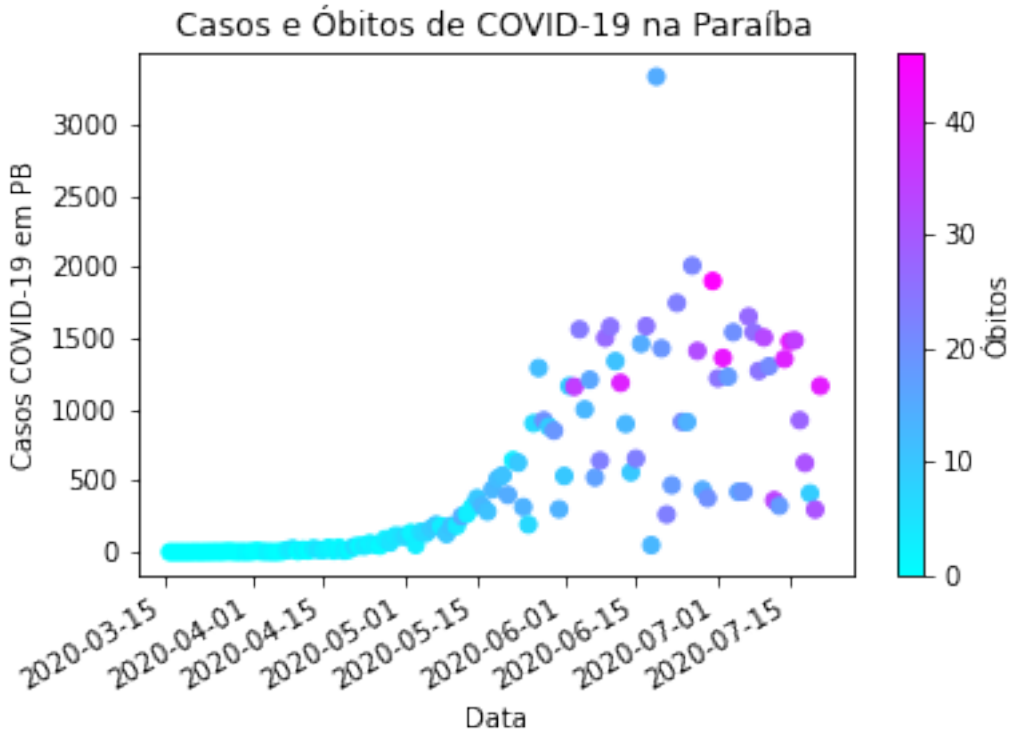


[31]:
```
fig, ax = plt.subplots()
ax.scatter(df_exemplo.index, df_exemplo['coluna_1'], s = np.
 ↪abs(df_exemplo['coluna_2'])*100)
fig.autofmt_xdate()
ax.set_xlabel('Data')
ax.set_ylabel('Valores da Coluna 1')
_ = ax.set_title('Gráfico do df_exemplo')
```

[32]:
```
covid_PB_casos_obitos = covid_PB[['obitosNovos', 'casosNovos']].sort_index()
fig, ax = plt.subplots()
grafico = ax.scatter(covid_PB_casos_obitos.index, covid_PB_casos_obitos.
 ↪casosNovos, c = covid_PB_casos_obitos.obitosNovos)
fig.autofmt_xdate()
ax.set_xlabel('Data')
ax.set_ylabel('Casos COVID-19 em PB')
ax.set_title('Casos e Óbitos de COVID-19 na Paraíba')
_ = plt.colorbar(grafico, label = 'Óbitos')
```

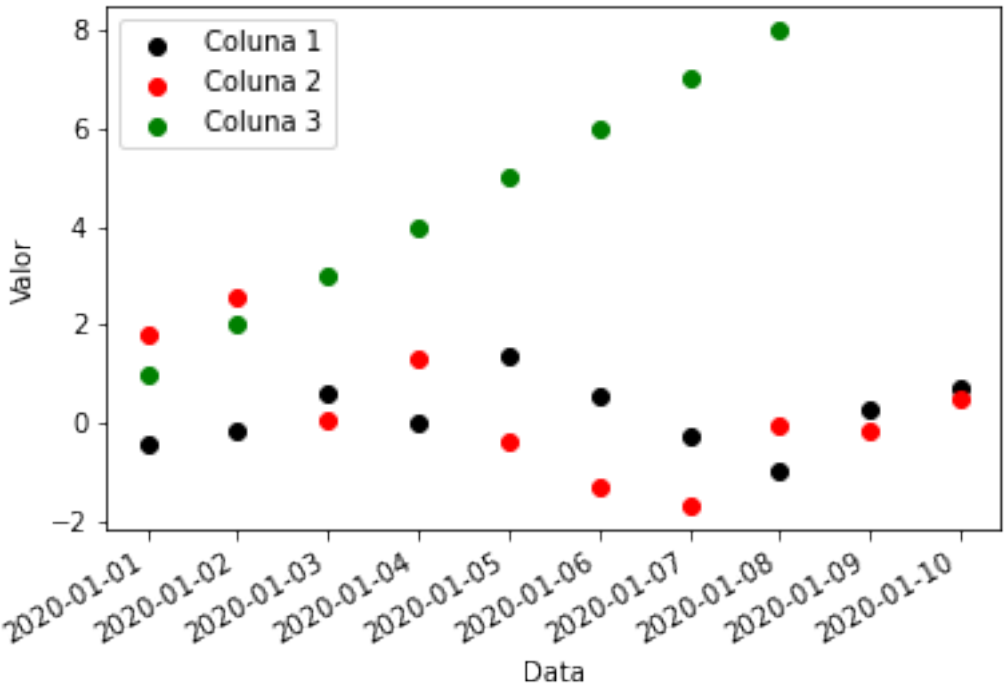


[33]:
```
covid_PB_casos_obitos = covid_PB[['obitosNovos', 'casosNovos']].sort_index()
fig, ax = plt.subplots()
grafico = ax.scatter(covid_PB_casos_obitos.index, covid_PB_casos_obitos.
 ↪casosNovos, c = covid_PB_casos_obitos.obitosNovos,
                     cmap='cool')
fig.autofmt_xdate()
ax.set_xlabel('Data')
ax.set_ylabel('Casos COVID-19 em PB')
ax.set_title('Casos e Óbitos de COVID-19 na Paraíba')
color_map=ax.get_children()[4]
_ = plt.colorbar(grafico, label = 'Óbitos')
```



[34]:
```
fig, ax = plt.subplots()
ax.scatter(df_exemplo.index, df_exemplo['coluna_1'], label = 'Coluna 1', color␣
 ↪= 'black')
ax.scatter(df_exemplo.index, df_exemplo['coluna_2'], label = 'Coluna 2', color␣
 ↪= 'red')
ax.scatter(df_exemplo.index, df_exemplo['coluna_3'], label = 'Coluna 3', color␣
 ↪= 'green')
fig.autofmt_xdate()
ax.legend()
ax.set_ylabel("Valor")
ax.set_xlabel("Data")
```

[34]:
```
Text(0.5, 0, 'Data')
```

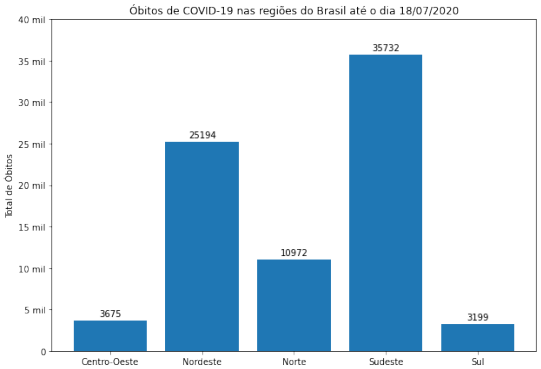


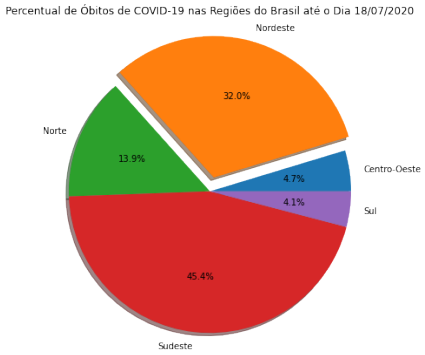
1.7 Gráficos Lado a Lado

[35]:
```
#Vamos modificar esta função para podermos utilizá-la quando temos mais de um␣
↪gráfico ao mesmo tempo
def autolabel(rects, ax):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        #ax.annotate('{}'.format(height), #antigo
        ax.annotate('{:.0f}'.format(height), #Modificamos para apresentar o␣
↪número inteiro
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')
```

[36]:
```
covid_Regioes = covid_BR[['regiao','obitosNovos']].groupby('regiao').sum().
↪query('regiao != "Brasil"')/2
figs, axs = plt.subplots(1,2, figsize=(22,7))
axs[0].set_ylim(0, 40000) # aumentamos o limite da coordenada y
retangulos = axs[0].bar(covid_Regioes.index, covid_Regioes['obitosNovos'])
axs[0].yaxis.set_major_formatter(FuncFormatter(inserir_mil))
axs[0].set_ylabel('Total de Óbitos') # Rótulo do eixo y
```

```
axs[0].set_title("Óbitos de COVID-19 nas regiões do Brasil até o dia 18/07/
↪2020")
autolabel(retangulos, axs[0])
axs[1].pie(covid_Regioes_pct.obitosNovos, explode=covid_Regioes_pct.explodir,
       labels=covid_Regioes_pct.index, autopct='%1.1f%%', shadow=True)
axs[1].set_title('Percentual de Óbitos de COVID-19 nas Regiões do Brasil até o␣
↪Dia 18/07/2020')
_ = axs[1].axis('equal')
```





1.8 Histograma

- Neste caso os comandos são praticamente os mesmos do método **plot** do *pandas*.

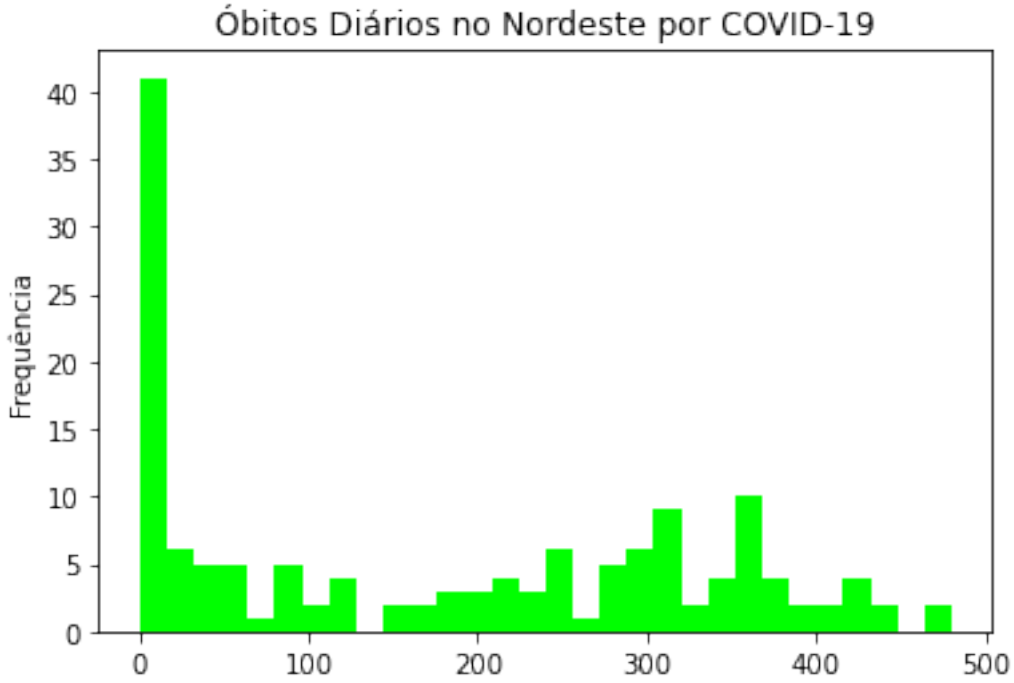
[37]:
```
covid_regioes_diarios = pd.DataFrame()

regioes = covid_BR.query('regiao != "Brasil"')['regiao'].drop_duplicates().array

for regiao in regioes:
    temp_series = covid_BR.set_index('data').query('regiao ==␣
↪@regiao')['obitosNovos'].groupby('data').sum()/2
    temp_series.name = 'obitos_' + regiao
    covid_regioes_diarios = pd.concat([covid_regioes_diarios, temp_series],␣
↪axis=1)

covid_regioes_diarios.index = pd.to_datetime(covid_regioes_diarios.index)
covid_regioes_diarios
```

[37]:

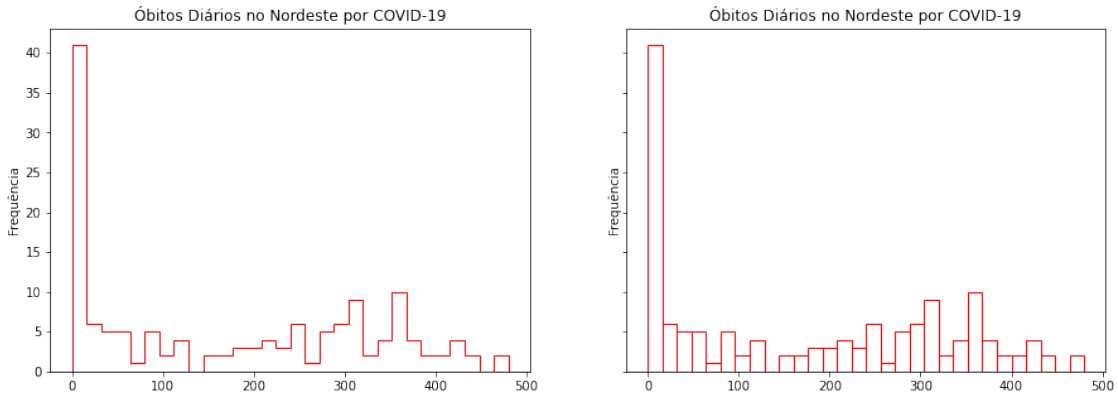
data	obitos_Norte	obitos_Nordeste	obitos_Sudeste	obitos_Sul
2020-02-25	0.0	0.0	0.0	0.0
2020-02-26	0.0	0.0	0.0	0.0
2020-02-27	0.0	0.0	0.0	0.0
2020-02-28	0.0	0.0	0.0	0.0

```
2020-02-29           0.0              0.0             0.0          0.0
…                    …                …               …            …
2020-07-14          76.0            315.0           682.0        139.0
2020-07-15          65.0            347.0           528.0        130.0
2020-07-16          97.0            373.0           611.0        105.0
2020-07-17         121.0            257.0           517.0        129.0
2020-07-18          61.0            292.0           358.0         95.0

            obitos_Centro-Oeste
data
2020-02-25                  0.0
2020-02-26                  0.0
2020-02-27                  0.0
2020-02-28                  0.0
2020-02-29                  0.0
…                           …
2020-07-14                 88.0
2020-07-15                163.0
2020-07-16                136.0
2020-07-17                139.0
2020-07-18                115.0

[145 rows x 5 columns]
```

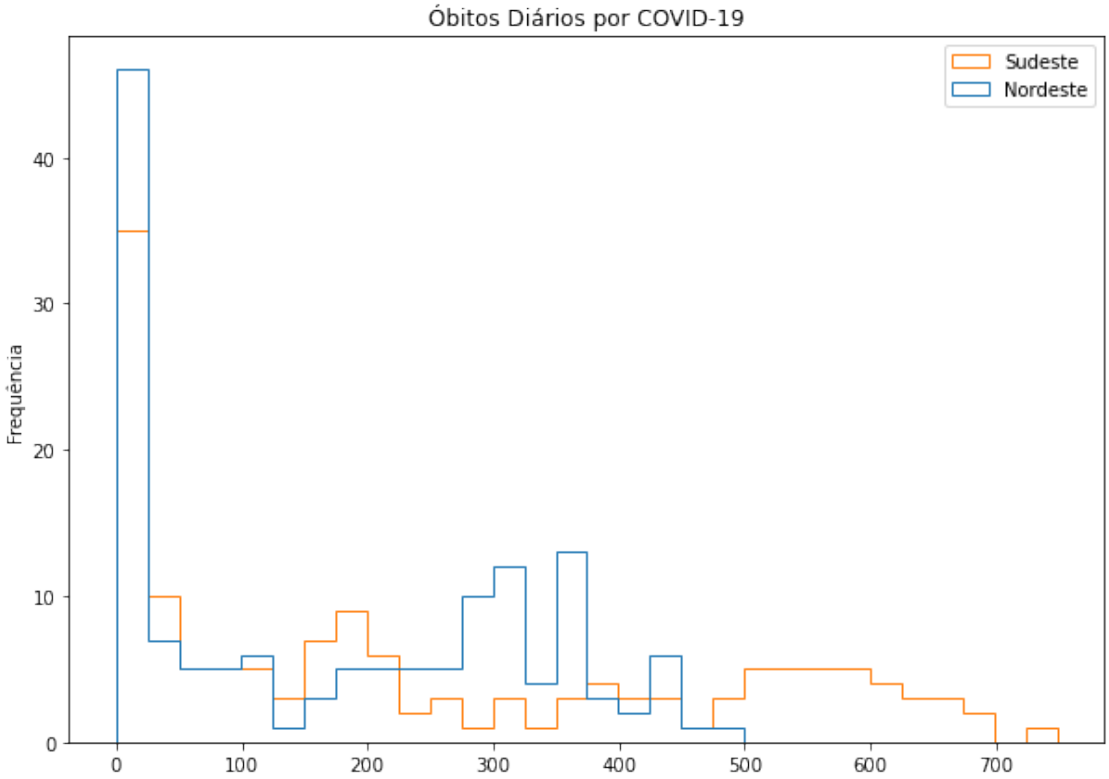
[38]:
```
fig, ax = plt.subplots()
ax.hist(covid_regioes_diarios.obitos_Nordeste, bins=30, color='lime')
ax.set_ylabel('Frequência')
_ = ax.set_title('Óbitos Diários no Nordeste por COVID-19')
```



[39]:
```
fig, axs = plt.subplots(1,2, sharey=True, figsize = (15,5)) #sharey=True indica␣
 ↪que o eixo y será o mesmo para todos os gráficos
axs[0].hist(covid_regioes_diarios.obitos_Nordeste, bins=30, histtype='step',␣
 ↪color='red')
axs[0].set_ylabel('Frequência')
axs[0].set_title('Óbitos Diários no Nordeste por COVID-19')
axs[1].hist(covid_regioes_diarios.obitos_Nordeste, bins=30, fill=False,␣
 ↪edgecolor='red')
axs[1].set_ylabel('Frequência')
_ = axs[1].set_title('Óbitos Diários no Nordeste por COVID-19')
```



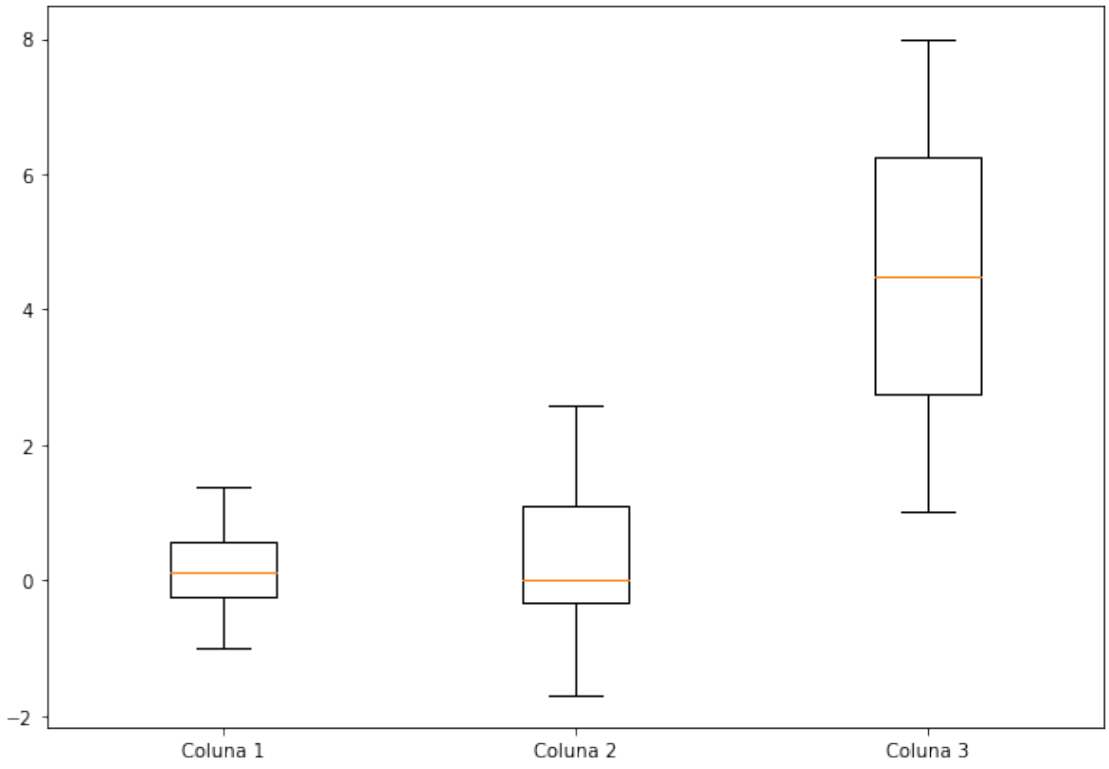
[40]:
```
fig, ax = plt.subplots(figsize=(10,7))
ax.hist([covid_regioes_diarios.obitos_Nordeste, covid_regioes_diarios.
 ↪obitos_Sudeste],
        bins=30, histtype='step', label=['Nordeste', 'Sudeste'])
ax.set_ylabel('Frequência')
ax.set_title('Óbitos Diários por COVID-19')
_ = ax.legend()
```



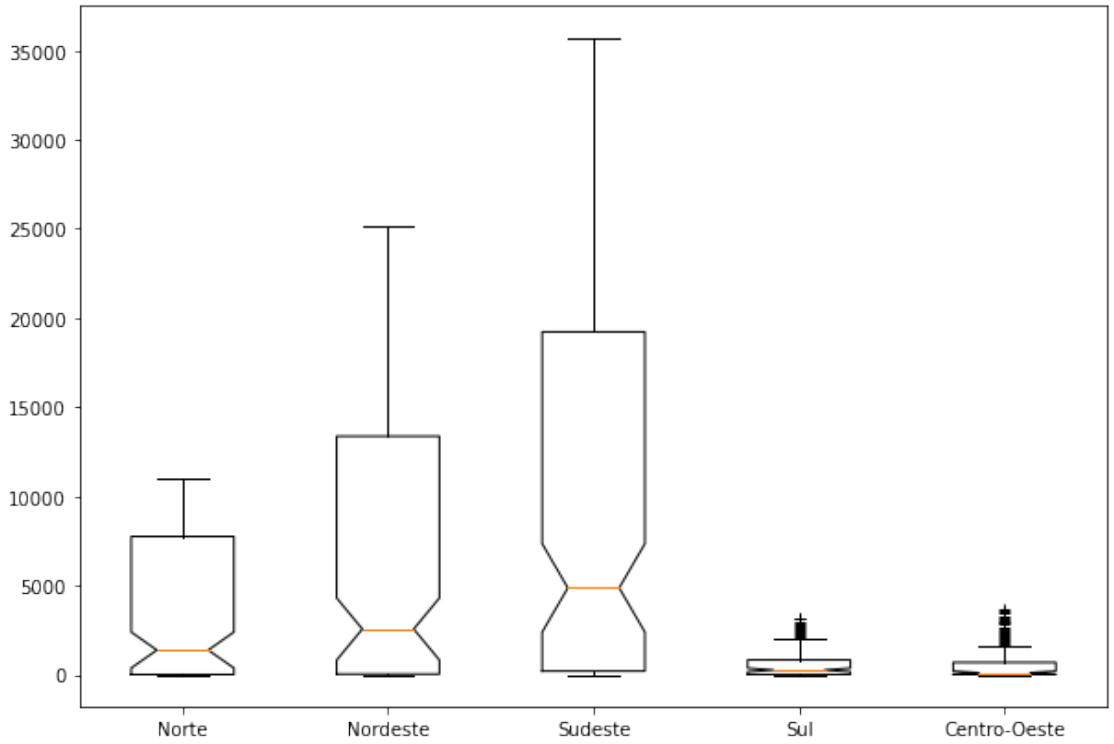
1.9 BoxPlot

O método para criar o *boxplot* utilizando o **matplotlib** se resume a fornecer uma lista (ou similar) de valores para os quais queremos os *boxplots* e uma lista (ou similar) contendo as posições nas quais queremos que os *boxplots* apareçam.

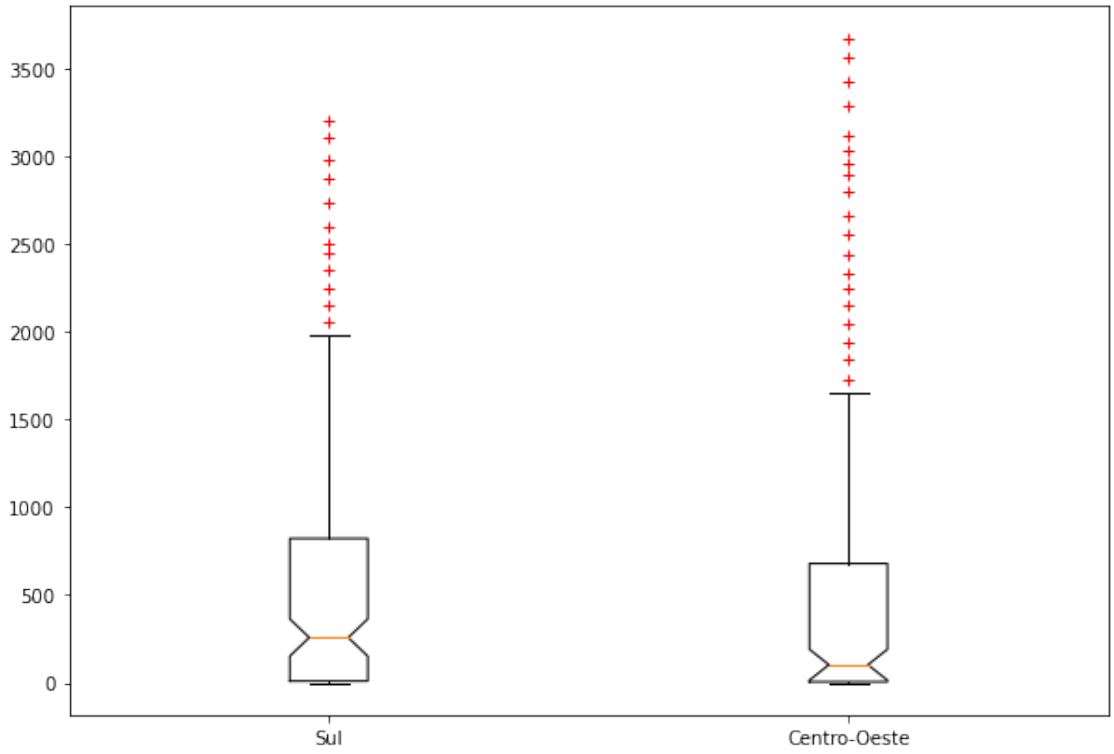
[41]:
```
fig, ax = plt.subplots(figsize=(10,7))
dados = [df_exemplo['coluna_1'], df_exemplo['coluna_2'], df_exemplo['coluna_3'].
 ↪dropna()]
posicoes = np.array(range(len(dados))) + 1
ax.boxplot(dados, positions=posicoes)
_ = ax.set_xticklabels(['Coluna 1', 'Coluna 2', 'Coluna 3'])
```



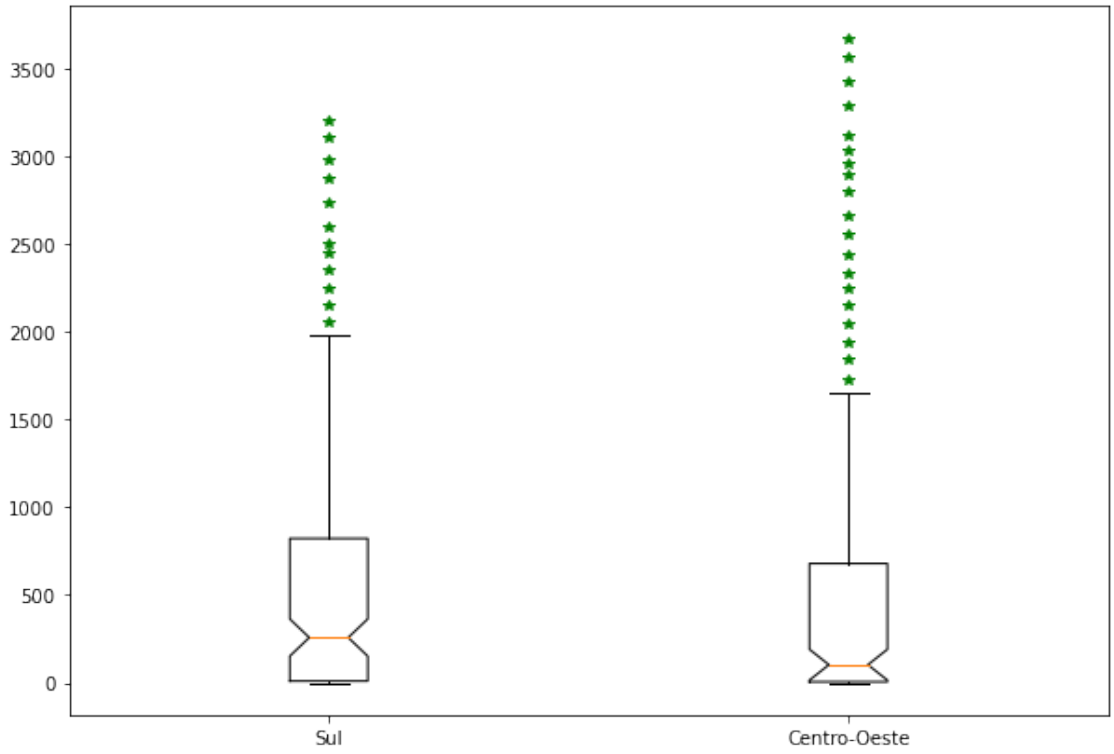
[42]:
```
covid_norte = covid_regioes.obitos_Norte
covid_nordeste = covid_regioes.obitos_Nordeste
covid_sudeste = covid_regioes.obitos_Sudeste
covid_sul = covid_regioes.obitos_Sul
covid_centro_oeste = covid_regioes['obitos_Centro-Oeste']
covid_box = [covid_norte, covid_nordeste, covid_sudeste, covid_sul,␣
 ↪covid_centro_oeste]
fig, ax = plt.subplots(figsize=(10,7))
posicoes = np.array(range(len(covid_box))) + 1
ax.boxplot(covid_box, 1, positions=posicoes, sym='+')
_ = ax.set_xticklabels(['Norte', 'Nordeste', 'Sudeste', 'Sul', 'Centro-Oeste'])
```



[43]:
```
covid_box_2 = [covid_sul, covid_centro_oeste]
fig, ax = plt.subplots(figsize=(10,7))
posicoes = np.array(range(len(covid_box_2))) + 1
ax.boxplot(covid_box_2, 1, positions=posicoes, sym='r+') #r indica 'red',␣
 ↪vermelho, + é o símbolo para o outlier
_ = ax.set_xticklabels(['Sul', 'Centro-Oeste'])
```



[44]:
```
covid_box_2 = [covid_sul, covid_centro_oeste]
fig, ax = plt.subplots(figsize=(10,7))
posicoes = np.array(range(len(covid_box_2))) + 1
ax.boxplot(covid_box_2, 1, positions=posicoes, sym='g*') #g indica 'green',␣
 ↪vermelho, * é o símbolo para o outlier
_ = ax.set_xticklabels(['Sul', 'Centro-Oeste'])
```



Importante: Muitos dos argumentos utilizados nos métodos apresentados também funcionam no método **plot** do *pandas*, vale a pena testar!

2 Criando gráficos interativos com o *plotly*

- **Plotly** é uma biblioteca de visualização de dados que permite a criação de gráficos interativos.
- Inicialmente utilizaremos um pacote rápido e eficiente para a construção de gráficos interativos: o **plotly.express**

[45]:
```
import plotly.express as px
```

2.1 Gráficos de linha

- Vamos começar criando gráficos de linha.
- Para este tipo de plot é conveniente ter apenas um valor possível para a coordenada y e ter uma segunda coluna determinando a cor a ser utilizada, diferenciando as regiões.
- Vamos então refazer nosso exemplo do Covid por regiões.

Preparando o banco de dados para o **plotly.express**:

[46]:
```
covid_regioes_px = covid_BR.query('regiao != "Brasil"')[['obitosAcumulado',␣
↪'regiao','data']].rename(
```

```
    {'obitosAcumulado':'Total de Óbitos','regiao':'Região','data':
↪'Data'},axis=1)
covid_regioes_px = covid_regioes_px.groupby(['Data','Região']).sum()/2
covid_regioes_px = covid_regioes_px.reset_index().set_index('Data')
```

[47]:
```
covid_regioes_px
```

[47]:
```
                 Região  Total de Óbitos
Data
2020-02-25  Centro-Oeste              0.0
2020-02-25      Nordeste              0.0
2020-02-25         Norte              0.0
2020-02-25       Sudeste              0.0
2020-02-25           Sul              0.0
…                    …                …
2020-07-18  Centro-Oeste           3675.0
2020-07-18      Nordeste          25194.0
2020-07-18         Norte          10972.0
2020-07-18       Sudeste          35732.0
2020-07-18           Sul           3199.0

[725 rows x 2 columns]
```

[48]:
```
fig = px.line(covid_regioes_px, y="Total de Óbitos", color="Região",
              line_group="Região", hover_name="Região", title='Óbitos de␣
↪COVID-19 nas regiões do Brasil')
fig.show()
```

- Podemos fixar o mesmo valor da coordenada x para todas as regiões na hora de passar o *mouse*.

[49]:
```
fig = px.line(covid_regioes_px, y="Total de Óbitos", color="Região",␣
↪line_group="Região", hover_name="Região", title='Óbitos de COVID-19 nas␣
↪regiões do Brasil')
fig.update_layout(hovermode='x unified')
fig.show()
```

- Vamos agora construir o mesmo gráfico com o pacote **plotly.graph_objects**.
- Não possui a simplicidade do **plotly.express**, porém possui mais flexibilidade e é mais "customizável".
- Para exemplificar a utilidade dele, vamos utilizar no conjunto de dados *covid_regioes* que possui 5 colunas distintas como valores de y.
- Além disso, veremos que o gráfico com x unificado ficará naturalmente melhor no **plotly.graph_objects**.
- Muitos argumentos disponíveis no **plotly.graph_objects** não estão disponíveis no **plotly.express**.

[50]:
```
import plotly.graph_objects as go
```

[51]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Norte'], mode='lines', name='Norte'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Nordeste'], mode='lines', name='Nordeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Centro-Oeste'], mode='lines', name='Centro-Oeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Sudeste'], mode='lines', name='Sudeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index, y=covid_regioes['obitos_Sul'],␣
 ↪mode='lines', name='Sul'))
fig.update_layout( title='Óbitos de COVID-19 nas regiões do Brasil',
                   xaxis_title='Data', yaxis_title='Total de Óbitos',␣
 ↪legend_title_text='Região', hovermode='x unified')
```

Vamos agora reordenar para melhor apresentação:

[52]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Sudeste'], mode='lines', name='Sudeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Nordeste'], mode='lines', name='Nordeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Norte'], mode='lines', name='Norte'))
fig.add_trace(go.Scatter(x=covid_regioes.index,␣
 ↪y=covid_regioes['obitos_Centro-Oeste'], mode='lines', name='Centro-Oeste'))
fig.add_trace(go.Scatter(x=covid_regioes.index, y=covid_regioes['obitos_Sul'],␣
 ↪mode='lines', name='Sul'))
fig.update_layout( title='Óbitos de COVID-19 nas regiões do Brasil',
                  xaxis_title='Data', yaxis_title='Total de Óbitos',␣
 ↪legend_title_text='Região', hovermode='x unified')
```

2.2 Gráficos de coluna

[53]:
```
fig = px.bar(covid_Regioes.reset_index().rename({'regiao':
 ↪'Região','obitosNovos':'Total de Óbitos'}, axis=1),
                                                x='Região', y='Total de Óbitos',
                                                title='Óbitos por COVID-19 nas␣
 ↪Regiões do Brasil')
fig.show()
```

- Utilizando o **graph_objects**:

[54]:
```
fig = go.Figure([go.Bar(x=covid_Regioes.index, y=covid_Regioes.obitosNovos)])
fig.update_layout( title='Óbitos de COVID-19 nas regiões do Brasil',
                   xaxis_title='Região', yaxis_title='Total de Óbitos')
fig.show()
```

[55]:
```
covid_coluna = covid_Regioes.reset_index().rename({'regiao':
 ↪'Região','obitosNovos':'Total de Óbitos'}, axis=1)
covid_coluna['Regiões'] = '' #Criamos uma coluna igual para todos para servir␣
 ↪de coordenada x
fig = px.bar(covid_coluna, x='Regiões', y='Total de Óbitos', color='Região',
             title='Óbitos por COVID-19 nas Regiões do Brasil até o dia 18/07/
 ↪2020',
             barmode='group') #Esse argumento coloca as colunas lado a lado
fig.show()
```

[56]:
```
fig = go.Figure(data=[
    go.Bar(name='Norte', x=['Óbitos'], y=covid_Regioes.loc['Norte']),
    go.Bar(name='Nordeste', x=['Óbitos'], y=covid_Regioes.loc['Nordeste']),
    go.Bar(name='Centro-Oeste', x=['Óbitos'], y=covid_Regioes.
 ↪loc['Centro-Oeste']),
    go.Bar(name='Sudeste', x=['Óbitos'], y=covid_Regioes.loc['Sudeste']),
    go.Bar(name='Sul', x=['Óbitos'], y=covid_Regioes.loc['Sul'])
])
fig.update_layout(barmode='group', title='Óbitos por COVID-19 nas Regiões do␣
 ↪Brasil',
                   yaxis_title='Total de Óbitos', legend_title_text='Região')
fig.update_xaxes(showticklabels=False)
fig.show()
```

[57]:
```
fig = px.bar(covid_coluna, x='Regiões', y='Total de Óbitos', color='Região',
                                              title='Óbitos por COVID-19 nas␣
 ↪Regiões do Brasil')
#Sem o argumento barmode='group' ficamos com as colunas empilhadas
fig.show()
```

[58]:
```
fig = go.Figure(data=[
    go.Bar(name='Norte', x=['Óbitos'], y=covid_Regioes.loc['Norte']),
    go.Bar(name='Nordeste', x=['Óbitos'], y=covid_Regioes.loc['Nordeste']),
    go.Bar(name='Centro-Oeste', x=['Óbitos'], y=covid_Regioes.
 ↪loc['Centro-Oeste']),
    go.Bar(name='Sudeste', x=['Óbitos'], y=covid_Regioes.loc['Sudeste']),
    go.Bar(name='Sul', x=['Óbitos'], y=covid_Regioes.loc['Sul'])
])
fig.update_layout(barmode='stack', title='Óbitos por COVID-19 nas Regiões do␣
 ↪Brasil',
                   yaxis_title='Total de Óbitos', legend_title_text='Região')
```

```
fig.update_xaxes(showticklabels=False)
fig.show()
```

2.3 Gráfico de Setores

O método *pie* das bibliotecas **plotly.express** e **plotly.graph_objects** são bastante imediatos e se assemelham muito ao que vimos anteriormente para o **matplotlib**.

[59]:
```
fig = px.pie(covid_Regioes, values='obitosNovos', names=covid_Regioes.index,
             title = 'Distribuição dos Óbitos por COVID-19 nas Regiões do␣
 ↪Brasil até 18/07/2020')
fig.show()
```

[60]:
```
fig = go.Figure(data=[go.Pie(labels=covid_Regioes_pct.index,␣
 ↪values=covid_Regioes_pct.obitosNovos,
                              pull=covid_Regioes_pct.explodir)])
fig.update_layout(title='Distribuição dos Óbitos por COVID-19 nas Regiões do␣
 ↪Brasil até 18/07/2020',
                  yaxis_title='Total de Óbitos', legend_title_text='Região')
fig.show()
```

2.4 Gráfico de Dispersão

Na prática os gráficos de linha e de dispersão são realizados com o mesmo método no **plotly.graph_objects**. Já no **plotly.express** é análogo ao método que vimos para o **matplotlib**.

[61]:
```
df_exemplo_px = pd.DataFrame(df_exemplo['coluna_1']).rename({'coluna_1':
 ↪'Valor'}, axis=1)
df_exemplo_px['Coluna'] = 'Coluna 1'
df_exemplo_px_temp = pd.DataFrame(df_exemplo['coluna_2']).rename({'coluna_2':
 ↪'Valor'}, axis=1)
df_exemplo_px_temp['Coluna'] = 'Coluna 2'
df_exemplo_px = pd.concat([df_exemplo_px, df_exemplo_px_temp])
df_exemplo_px_temp = pd.DataFrame(df_exemplo['coluna_3']).rename({'coluna_3':
 ↪'Valor'}, axis=1)
df_exemplo_px_temp['Coluna'] = 'Coluna 3'
df_exemplo_px = pd.concat([df_exemplo_px, df_exemplo_px_temp])
df_exemplo_px.head()
```

[61]:

	Valor	Coluna
2020-01-01	-0.416092	Coluna 1
2020-01-02	-0.137970	Coluna 1
2020-01-03	0.575827	Coluna 1
2020-01-04	-0.017367	Coluna 1
2020-01-05	1.384279	Coluna 1

[62]:
```
fig = px.scatter(df_exemplo_px, x=df_exemplo_px.index, y='Valor',␣
↪color='Coluna')
fig.show()
```

Obs.: Utilizando o pacote podemos trabalhar diretamente com o *df_exemplo*.

[63]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_exemplo.index, y=df_exemplo['coluna_1'],␣
↪mode='markers', name='Coluna 1'))
fig.add_trace(go.Scatter(x=df_exemplo.index, y=df_exemplo['coluna_2'],␣
↪mode='markers',name='Coluna 2'))
fig.add_trace(go.Scatter(x=df_exemplo.index, y=df_exemplo['coluna_3'],␣
↪mode='markers',name='Coluna 3'))
fig.update_layout( title='Gráfico de Dispersão do df_exemplo',
xaxis_title='Data', yaxis_title='Valor', legend_title_text='Coluna')
```

2.5 Histograma

Com o **plotly.express** podemos aplicar o método diretamente com poucas diferenças entre os argumentos. Vemos que no lugar de *bins*, devemos utilizar *nbins* e no lugar de *alpha*, devemos combinar *barmode='overlay'* com *opacity*.

[64]:
```
fig = px.histogram(covid_regioes_diarios.obitos_Nordeste, nbins=30, title='''
Distribuição da quantidade de óbitos diários de COVID-19 no nordeste do Brasil
                    ''')
fig.show()
```

[65]:
```
covid_regioes_diarios_px = covid_BR.query('regiao != "Brasil"')[['obitosNovos',␣
↪'regiao','data']].rename(
    {'obitosNovos':'Óbitos','regiao':'Região','data':'Data'},axis=1)
covid_regioes_diarios_px = covid_regioes_diarios_px.groupby(['Região','Data']).
↪sum()/2
covid_regioes_diarios_px = covid_regioes_diarios_px.reset_index().
↪set_index('Data')
covid_regioes_diarios_px
```

[65]:

Data	Região	Óbitos
2020-02-25	Centro-Oeste	0.0
2020-02-26	Centro-Oeste	0.0
2020-02-27	Centro-Oeste	0.0
2020-02-28	Centro-Oeste	0.0
2020-02-29	Centro-Oeste	0.0
…	…	…
2020-07-14	Sul	139.0
2020-07-15	Sul	130.0
2020-07-16	Sul	105.0

```
2020-07-17           Sul   129.0
2020-07-18           Sul    95.0

[725 rows x 2 columns]
```

[66]:
```
fig = px.histogram(covid_regioes_diarios_px, nbins=30, color='Região',␣
↪opacity=0.5, barmode='overlay', title='''
Distribuição da quantidade de óbitos diários de COVID-19 nas regiões do Brasil
                  ''')
fig.show()
```

Agora vejamos com **plotly.graph_objects**:

[67]:
```
def fazer_histograma_plotly():
    fig = go.Figure()
    fig.update_layout(barmode='overlay', title='''
Distribuição da quantidade de óbitos diários de COVID-19 nas regiões do Brasil
                  ''',
                      yaxis_title="Quantidade de Dias",␣
↪xaxis_title="Óbitos",legend_title_text='Região')
    fig.add_trace(go.Histogram(x=covid_regioes_diarios['obitos_Norte'],␣
↪name='Norte'))
    fig.add_trace(go.Histogram(x=covid_regioes_diarios['obitos_Nordeste'],␣
↪name='Nordeste'))
    fig.add_trace(go.Histogram(x=covid_regioes_diarios['obitos_Centro-Oeste'], ␣
↪name='Centro-Oeste'))
    fig.add_trace(go.Histogram(x=covid_regioes_diarios['obitos_Sudeste'],␣
↪name='Sudeste'))
    fig.add_trace(go.Histogram(x=covid_regioes_diarios['obitos_Sul'], ␣
↪name='Sul'))
    fig.update_traces(opacity=0.5, xbins={'size':50})
    fig.show()
```

[68]:
```
fazer_histograma_plotly()
```

2.6 BoxPlot

No **plotly** os argumentos do *boxplot* são muito semelhantes aos do histograma.

[69]:
```
fig = px.box(df_exemplo_px, x="Coluna", y="Valor")
fig.show()
```

[70]:
```
fig = px.box(covid_regioes_diarios_px, x="Região", y="Óbitos")
fig.show()
```

[71]:
```

```

```
fig = px.box(covid_regioes_diarios_px, x='Região', y='Óbitos', notched=True,␣
↪color='Região',
             title='Distribuição da quantidade de óbitos diários de COVID-19 nas␣
↪regiões do Brasil')
fig.show()
```

[72]:
```
def fazer_boxplot_plotly():
    fig = go.Figure()
    fig.update_layout(title='''
Distribuição da quantidade de óbitos diários de COVID-19 nas regiões do Brasil
                    ''',
                      yaxis_title="Óbitos", legend_title_text='Região')
    fig.add_trace(go.Box(y=covid_regioes_diarios['obitos_Norte'], name='Norte'))
    fig.add_trace(go.Box(y=covid_regioes_diarios['obitos_Nordeste'],␣
↪name='Nordeste'))
    fig.add_trace(go.Box(y=covid_regioes_diarios['obitos_Centro-Oeste'], ␣
↪name='Centro-Oeste'))
    fig.add_trace(go.Box(y=covid_regioes_diarios['obitos_Sudeste'],␣
↪name='Sudeste'))
    fig.add_trace(go.Box(y=covid_regioes_diarios['obitos_Sul'],  name='Sul'))
    fig.show()
```

[73]:
```
fazer_boxplot_plotly()
```