Programação II
+
Estruturas de Dados para Bioinformática

Gráficos interativos (matplotlib)

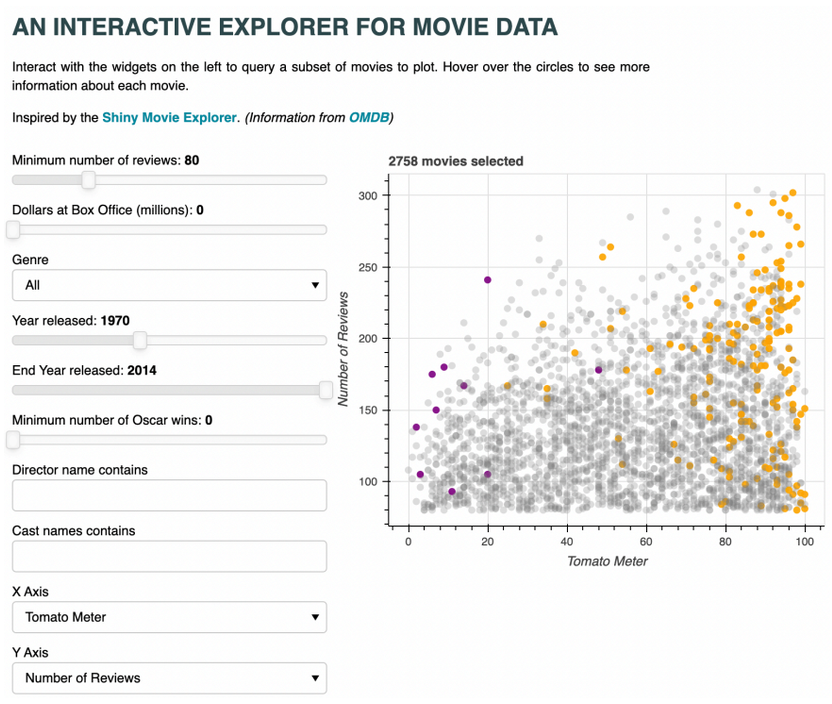
Hugo Pacheco

DCC/FCUP
22/23

Gráficos

- Últimas aulas:
  - Gráficos estáticos em *matplotlib*
    - Visualizar sempre a mesma informação
      - Exportar para imagem
- Esta aula:
  - Gráficos interativos em *matplotlib*
    - Alterar a visualização consoante input do utilizador: botões, barras, …
    - Utilizar a GUI para modo interativo
- Não vamos ver agora: outras bibliotecas para gráficos interativos (Bokeh, *plotly*, …)

Bokeh web example



Gráficos interativos

- Em *matplotlib*, um gráfico interativo consiste em:
  - Gráficos estáticos desenhados normalmente
  - Widgets que respondem a eventos
- Eventos podem:
    - Mostrar/esconder gráficos já desenhados
    - Alterar componentes dos gráficos (cor, eixos, …)

Matplotlib (*Axes*)

- Um gráfico em *matplotlib* é um objeto da classe *Axes*
- Algumas funções criam explicitamente *Axes*, e.g.

```
# criar um gráfico vazio
_,ax = plt.subplots()
# desenha uma Series e retorna um novo gráfico
ax = series.plot()
# desenha um DataFrame e retorna um novo gráfico
ax = dataframe.plot()
```

- Com métodos *get_atributo* e *set_atributo*, podemos alterar um *atributo* de um gráfico (lista completa):
  - *visible, xlim, ylim, xticks, yticks, xlabel, ylabel*, …

Matplotlib (*Line2D*)

- Um desenho de uma curva num gráfico é um objeto da classe *Line2D*

- A função *plt.plot(…)* cria explicitamente uma lista de *Line2D*, e.g.

```
# criar um gráfico vazio
_,ax = plt.subplots()
# desenha um plot no gráfico ax
ls1 = ax.plot(...)
# desenha um plot num novo gráfico
ls2 = plt.plot(...)
```

- Com métodos *get_atributo* e *set_atributo*, podemos alterar um *atributo* de uma *Line2D* (lista completa):

  - *visible, label*, *xdata, ydata, linestyle, linewidth, marker,* …

Widgets

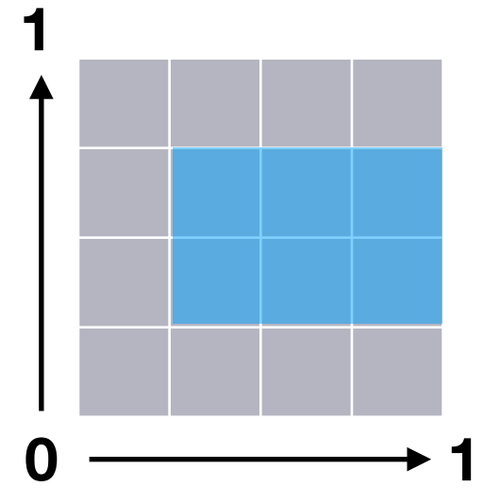
- Alguns widgets *matplotlib* que vamos ver:
    - Botões
    - Botões de seleção
    - Botões de seleção exclusiva
    - Barras deslizantes

Botões

- Um botão é um objeto retangular com texto e um evento “fui clickado”

- Um retângulo é construído como uma lista com 4 elementos entre 0 e 1 relativos à janela:
  - *[posicaoX,posicaoY,comprimentoX,comprimentoY]*
  - 0 = esquerda/baixo e 1 = direita/cima

[0.25,0.25,0.75,0.5]



- Um evento é uma função que recebe informação sobre o evento (e.g. a posição do rato) e não retorna nada, mas altera alguma propriedade dos gráficos como efeito secundário

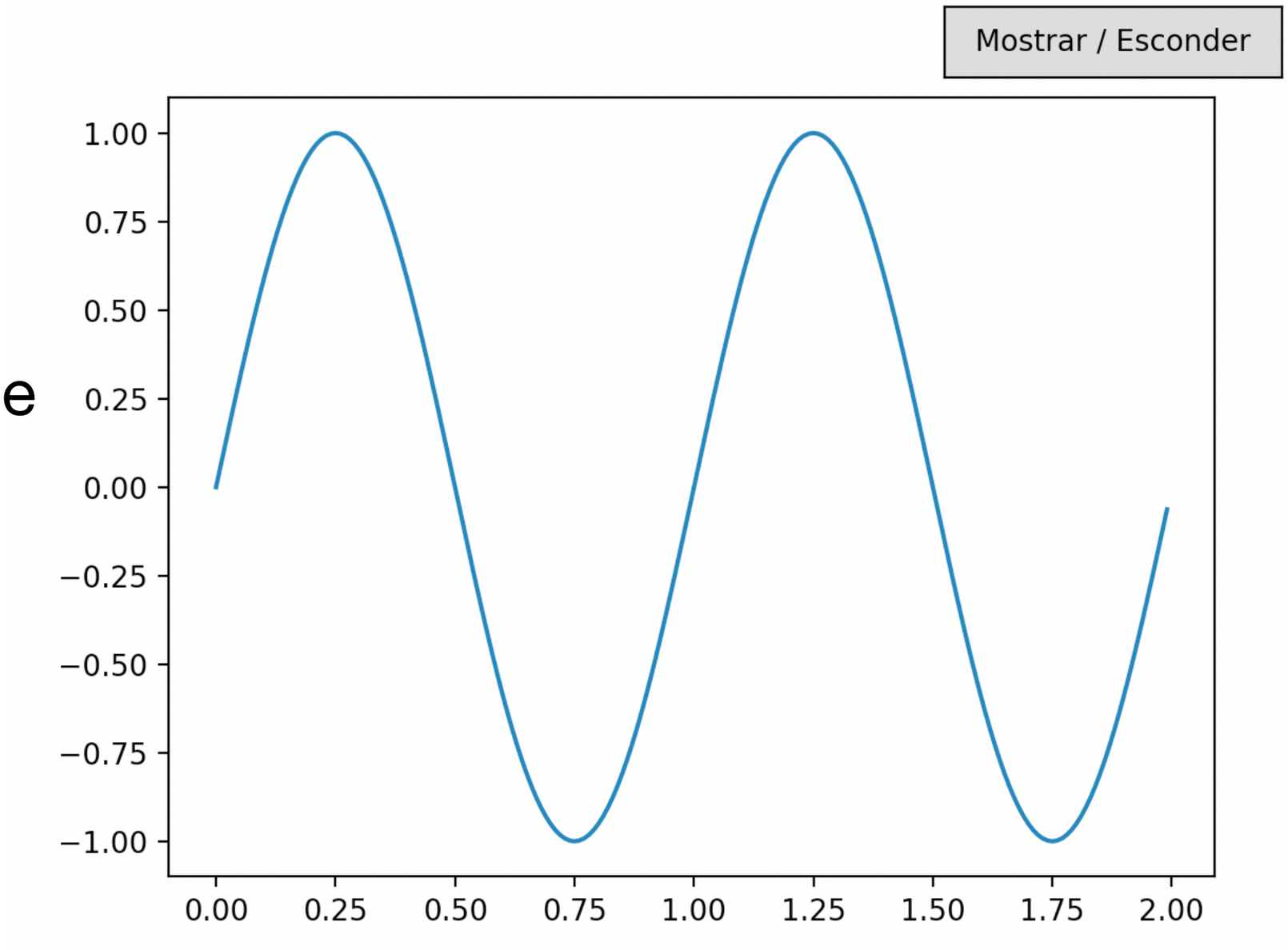
Botões

- E.g., um botão que mostra/esconde uma curva
- Evento
- Redesenha

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Button

# calcula um seno
xx = np.arange(0.0, 2.0, 0.01)
yy = np.sin(2*np.pi*xx)

# desenha uma linha, ln é o objeto
ln, = plt.plot(xx,yy)

# um retângulo perto do canto superior direito
rect = plt.axes([0.7, 0.9, 0.25, 0.07])
# um botão
button = Button(rect,'Mostrar / Esconder')

# a função que reage ao evento
def reage(info):
    # lê e altera a visibilidade do gráfico
    ln.set_visible(not ln.get_visible())
    plt.draw()

# liga o evento à função
button.on_clicked(reage)
plt.show()
```

Botões

- E.g., um botão que mostra/esconde uma curva



Botão de seleção

- Um botão de seleção (*check button*) é um objeto retangular com uma sequência de opções que podem ser ativadas/desativadas

  - Recebe um parâmetro opcional para definir o estado inicial de cada opção

- É como se cada opção fosse um botão independente

- O evento “fui clickado” diz qual a opção “clickada”

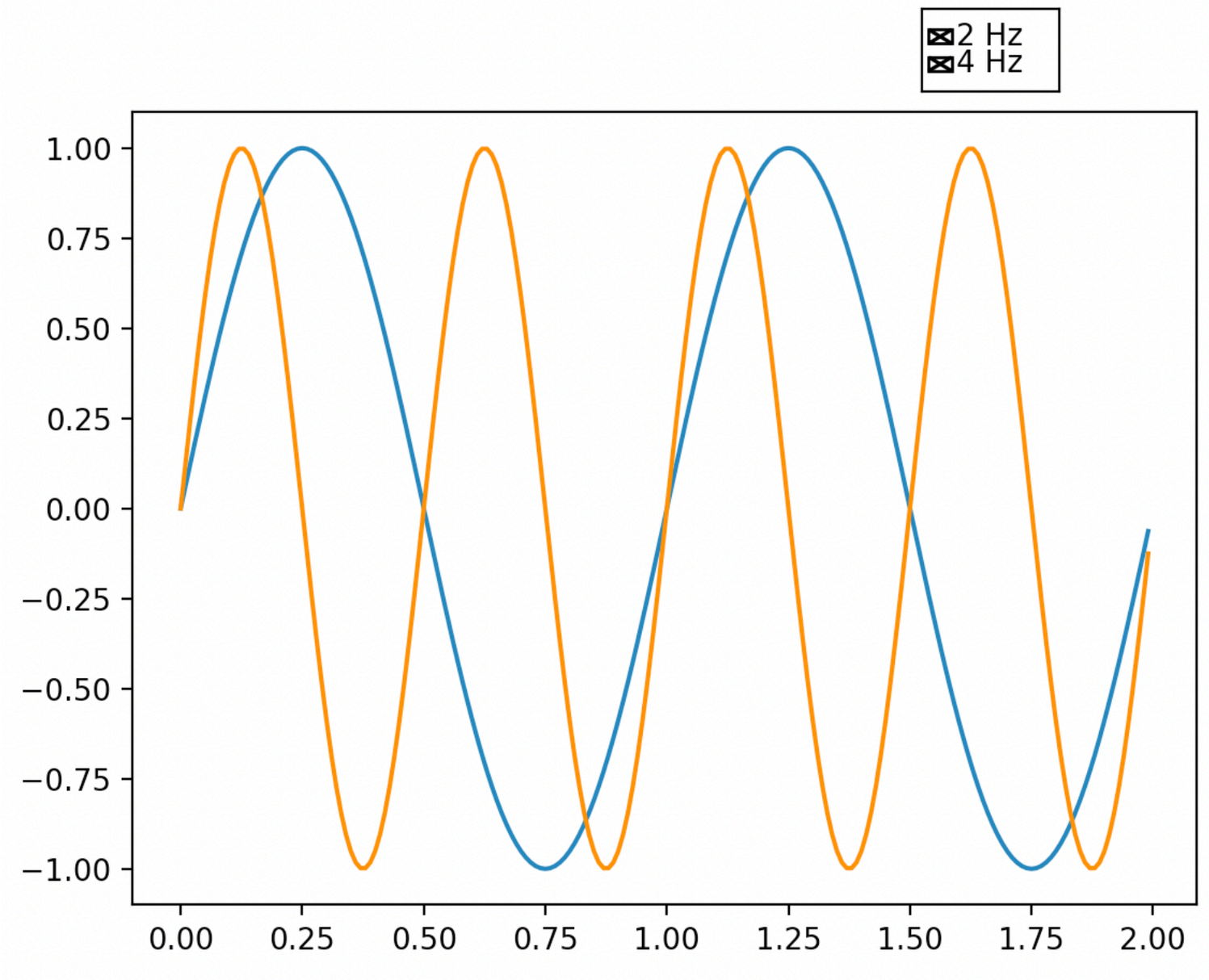
Botão de seleção

- E.g., um *check button* que permite mostrar/ esconder múltiplas curvas

```
from matplotlib.widgets import CheckButtons

xx = np.arange(0.0, 2.0, 0.01)
yy1 = np.sin(2*np.pi*xx)
yy2 = np.sin(4*np.pi*xx)

ln1, = plt.plot(xx,yy1)
ln2, = plt.plot(xx,yy2)

rect = plt.axes([0.7,0.9,0.1,0.08])
# um botão de seleção
button = CheckButtons(rect,('2 Hz','4 Hz'),
(True,True))

def reage_ln(ln):
    ln.set_visible(not ln.get_visible())
    plt.draw()

# a função que reage ao evento recebe a opção
def reage(label):
    if label=='2 Hz': reage_ax(ln1)
    elif label=='4 Hz': reage_ax(ln2)

button.on_clicked(reage)
plt.show()
```

Botão de seleção

- E.g., um *check button* que permite mostrar/ esconder múltiplas curvas



Botão de seleção exclusiva

- Um botão de seleção exclusiva (*radio button*) é um objeto retangular com uma sequência de opções em que uma e apenas uma está ativada

- O evento “fui clickado” diz qual a opção ativa de momento

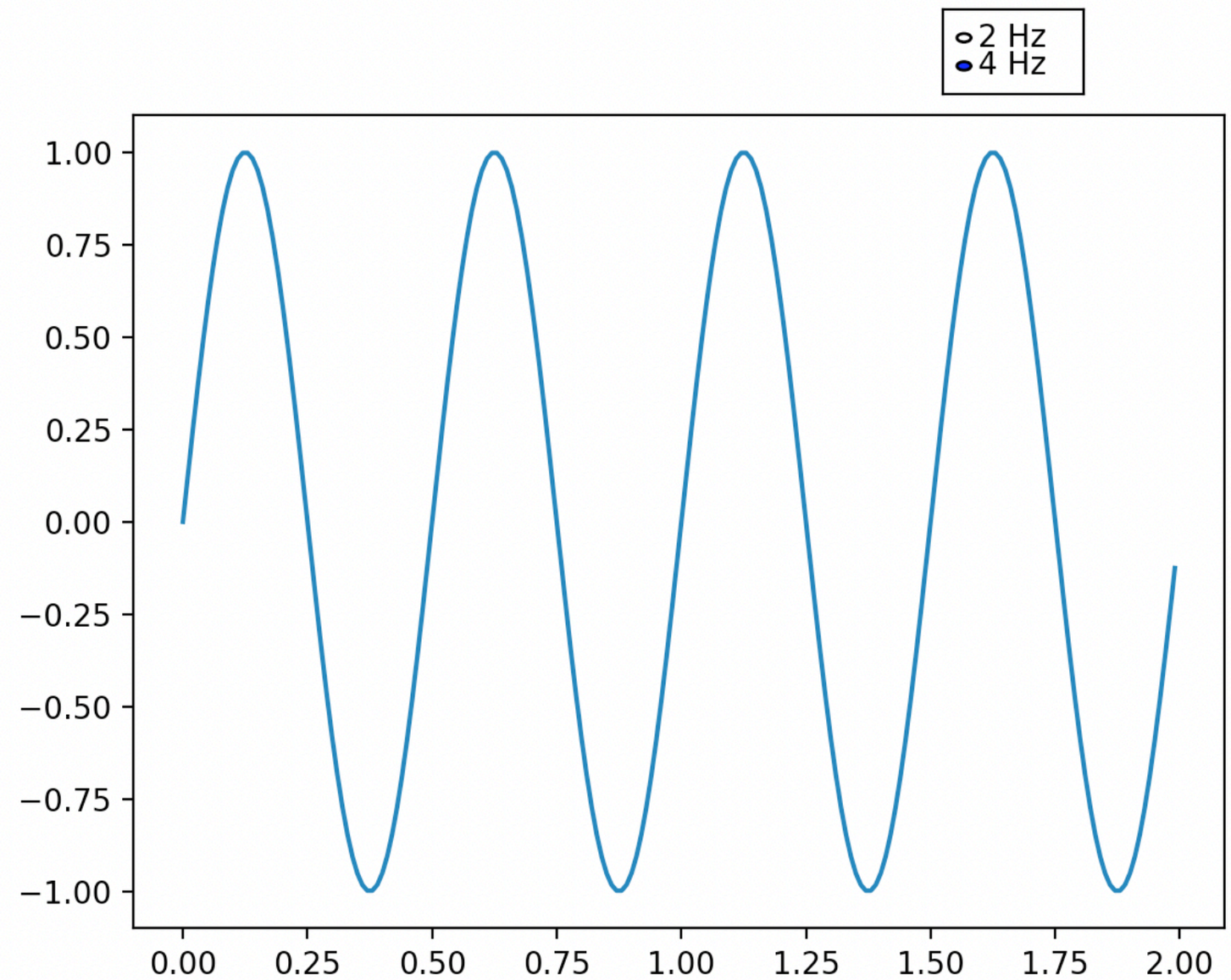
Botão de seleção exclusiva

- E.g., um *radio button* que permite alterar a frequência da curva

```
from matplotlib.widgets import RadioButtons

xx = np.arange(0.0, 2.0, 0.01)
yy = np.sin(2*np.pi*xx)

ln, = plt.plot(xx,yy)

rect = plt.axes([0.7,0.9,0.1,0.08])
# um botão de seleção exclusiva
button = RadioButtons(rect,('2 Hz','4 Hz'))

def reage_ln(n):
    # altera a curva desenhada
    ln.set_ydata(np.sin(n*np.pi*xx))
    plt.draw()

# a função que reage ao evento recebe a opção
def reage(label):
    if label=='2 Hz': reage_ln(2)
    elif label=='4 Hz': reage_ln(4)

button.on_clicked(reage)
plt.show()
```

Botão de seleção exclusiva

- E.g., um *radio button* que permite alterar a frequência da curva



Barra deslizante

- Uma barra deslizante (*slider*) é um objeto retangular com um nome e um intervalo de valores possíveis
  - Recebe parâmetros opcionais como valor inicial e “salto” mínimo

- O evento “fui alterada” diz qual o valor atual (dentro do intervalo)

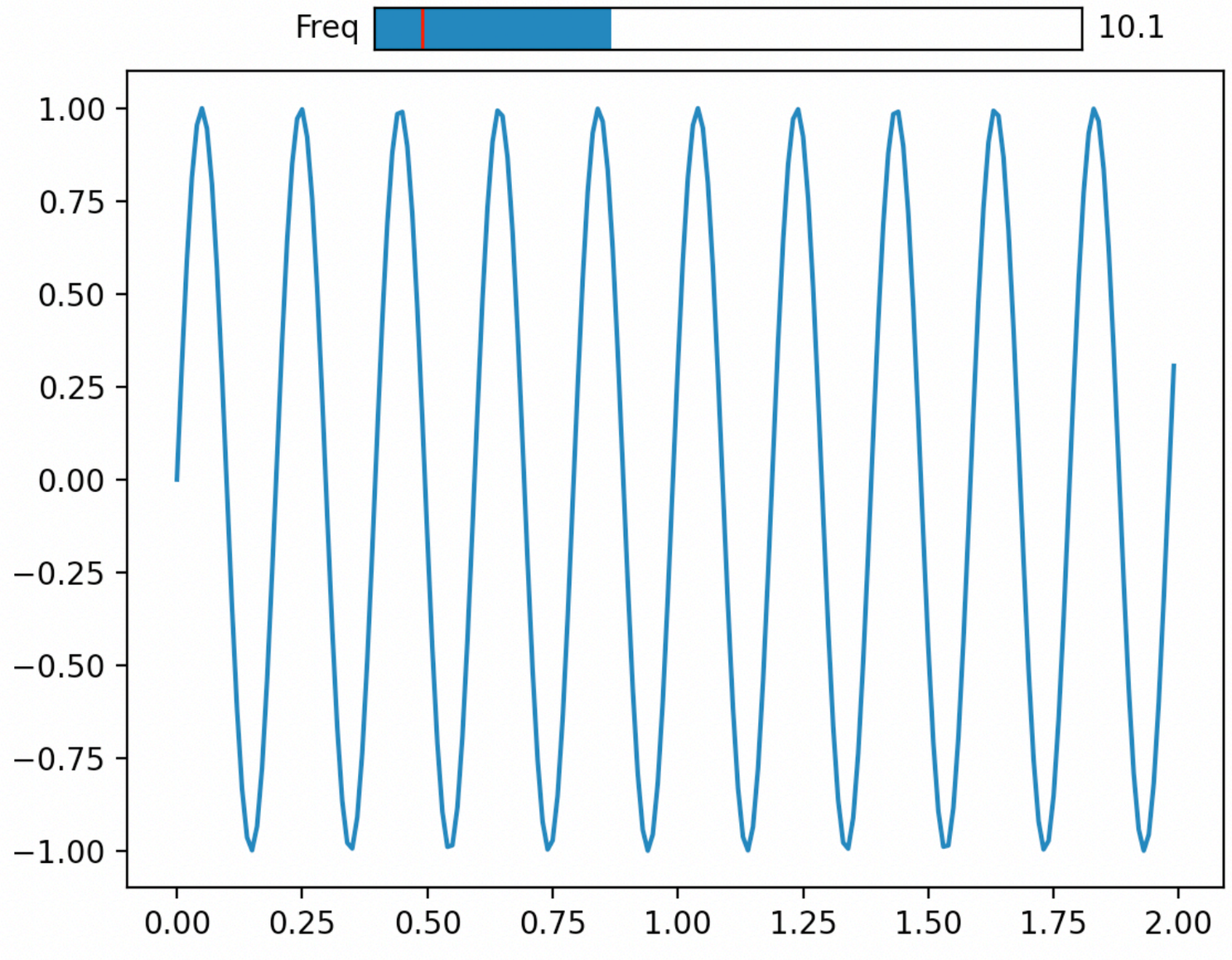
Barra deslizante

- E.g., um *slider* que permite alterar a frequência da curva

```
from matplotlib.widgets import Slider

xx = np.arange(0.0, 2.0, 0.01)
yy = np.sin(2*np.pi*xx)

line, = plt.plot(xx,yy)
rect = plt.axes([0.3,0.9,0.5,0.04])
# um slider
slider = Slider(rect, 'Freq', 0.1, 30.0,
valinit=2, valstep=0.5)

# a função que reage ao evento recebe o
novo valor
def reage(freq):
    # altera a curva desenhada
    line.set_ydata(np.sin(freq*np.pi*xx))
    plt.draw()

slider.on_changed(reage)
plt.show()
```

Barra deslizante

- E.g., um *slider* que permite alterar a frequência da curva



Exemplo 1

- Dados climatéricos anuais de longa duração fornecidos pelo IPMA aqui.
- Gráfico interativo das temperaturas mínima e máxima por mês, ano a ano

```
dfs = pd.read_excel("PT100-tx-tn-prec.xlsx", sheet_name=None)
months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct',
'Nov','Dec']
ymonths = ['year'] + months

tmins = dfs['tmin']
tmins.rename(columns={'Dez':'Dec'},inplace=True)
tmins = tmins[ymonths].copy()
tmins.dropna(inplace=True)
tmins['year'] = tmins['year'].astype('uint16')
tmins.set_index('year',inplace=True)

tmaxs = dfs['tmax']
tmaxs = tmaxs[ymonths].copy()
tmaxs.dropna(inplace=True)
tmaxs['year'] = tmaxs['year'].astype('uint16')
tmaxs.set_index('year',inplace=True)
```

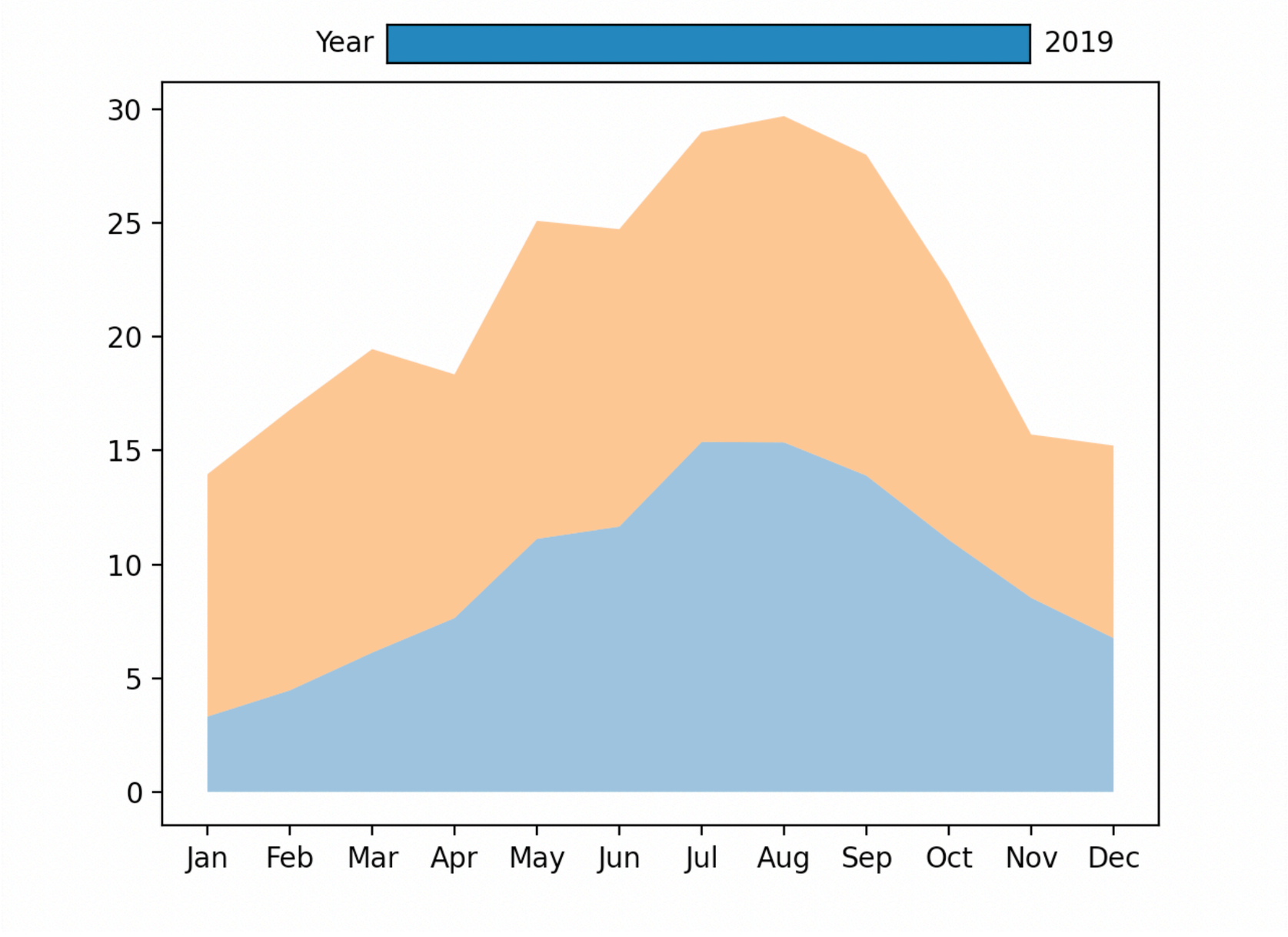
Exemplo 1

- Desenhar o gráfico com um *slider* anual

```
minyear = tmins.index[0]
maxyear = tmins.index[-1]
# cria um gráfico
_,ax = plt.subplots()
# desenha um ano
def draw_year(year):
    ax.fill_between(months,tmins.loc[year],alpha=0.5)
    ax.fill_between(months,tmins.loc[year],tmaxs.loc[year],alpha=0.5)
# desenha o ano mais recente por defeito
draw_year(maxyear)

# cria um novo slider
rect = plt.axes([0.3,0.9,0.5,0.04])
slider = Slider(rect,'Year',minyear,maxyear, valinit=maxyear, valstep=1)
# redesenha o gráfico para o ano atual
def reage(year):
    ax.clear()
    draw_year(year)
slider.on_changed(reage)
plt.show()
```

Exemplo 1

- Desenhar o gráfico com um *slider* anual



Exemplo 1

- Acrescentar visualização de precipitação

```
#carrega dados precipitação
precs = dfs['prec']
precs = precs[ymonths].copy()
precs.dropna(inplace=True)
precs['year'] = precs['year'].astype('uint16')
precs.set_index('year',inplace=True)

# cria um num novo eixo dos Y
ax2 = ax.twinx()
# desenha precipitação
lprec, = ax2.plot(months,precs.loc[maxyear],color='green')
# redesenha precipitação
def redraw_prec(year):
    lprec.set_ydata(precs.loc[year])
    ax2.set_ylim(0,precs.loc[year].max()+10)
# atualiza também precipitação
def reage(year):
    ...
    redraw_prec(year)
```

Exemplo 1

- Acrescentar *check button*

```
# cria dois eixos nos Y
ax.set_yticks([])
# renomear ax para ax1 no resto do código
ax1 = ax.twinx()
ax2 = ax1.twinx()

# um botão de seleção
rect = plt.axes([0.15,0.77,0.1,0.08])
button = CheckButtons(rect,('Temp','Prec'),(True,True))
def altera(label):
    if label=='Temp': ax1.set_visible(not ax1.get_visible())
    elif label=='Prec': ax2.set_visible(not ax2.get_visible())
    plt.draw()
button.on_clicked(altera)
```

Exemplo 1

- Gráfico de temperatura e precipitação anual