

Programação II

Gráficos

Parte 2

Hugo Pacheco

DCC/FCUP

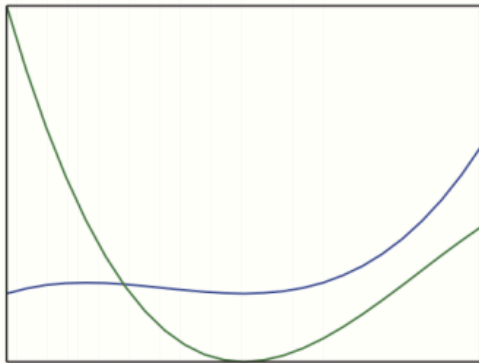
21/22

Gráficos

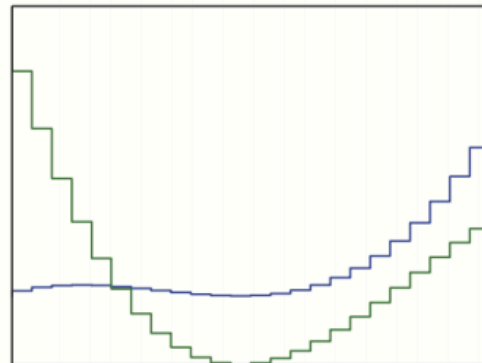
- Última aula
 - Introdução a gráficos no *matplotlib*
 - Tipos de gráficos *plot* e *imshow*
 - Customização de gráficos no *matplotlib*
- Esta aula:
 - Outros tipos de gráficos
 - Exemplos práticos

Tipos de gráficos

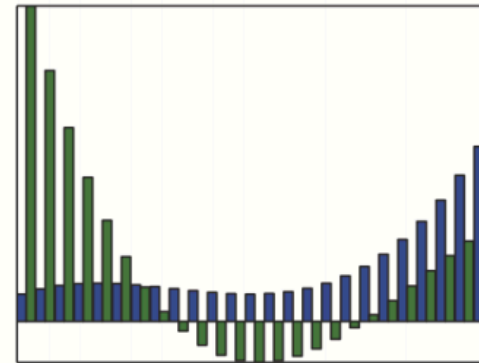
`Axes.plot`



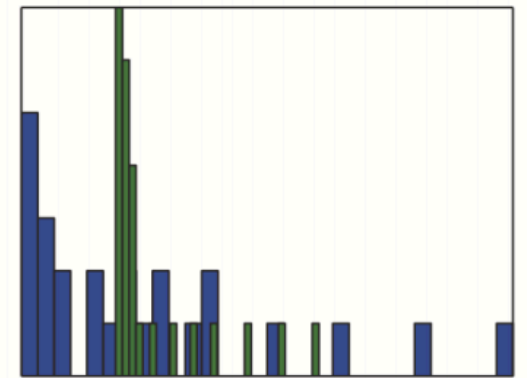
`Axes.step`



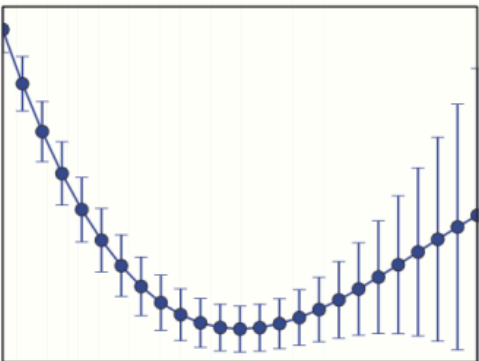
`Axes.bar`



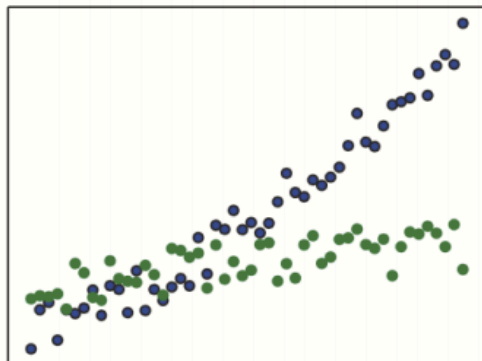
`Axes.hist`



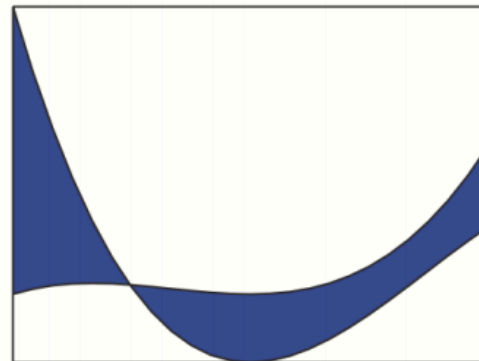
`Axes.errorbar`



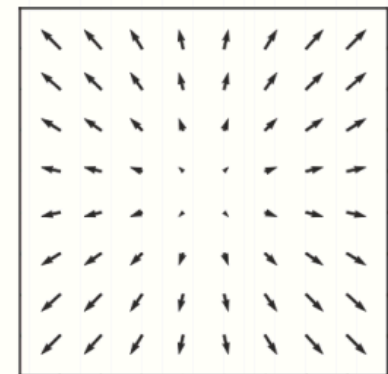
`Axes.scatter`



`Axes.fill_between`



`Axes.quiver`

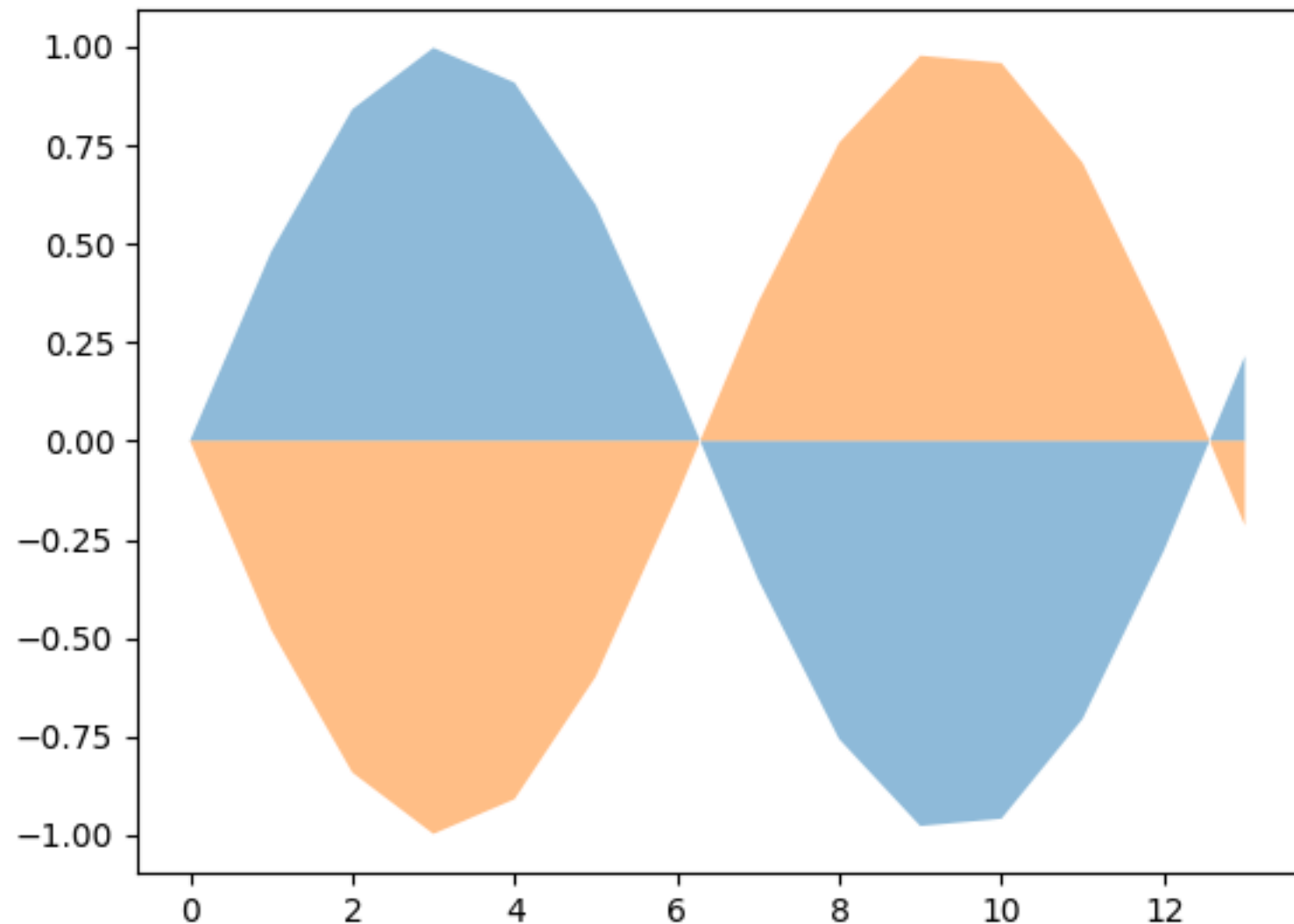


Matplotlib (fill)

```
x = np.arange(14)
y1 = np.sin(x / 2)
y2 = np.cos(math.pi/2 + x / 2)

plt.fill_between(x, y1, alpha=0.5) # blue
plt.fill_between(x, y2, alpha=0.5) # orange
plt.show()
```

- Preencher a área entre o eixo dos X e cada curva

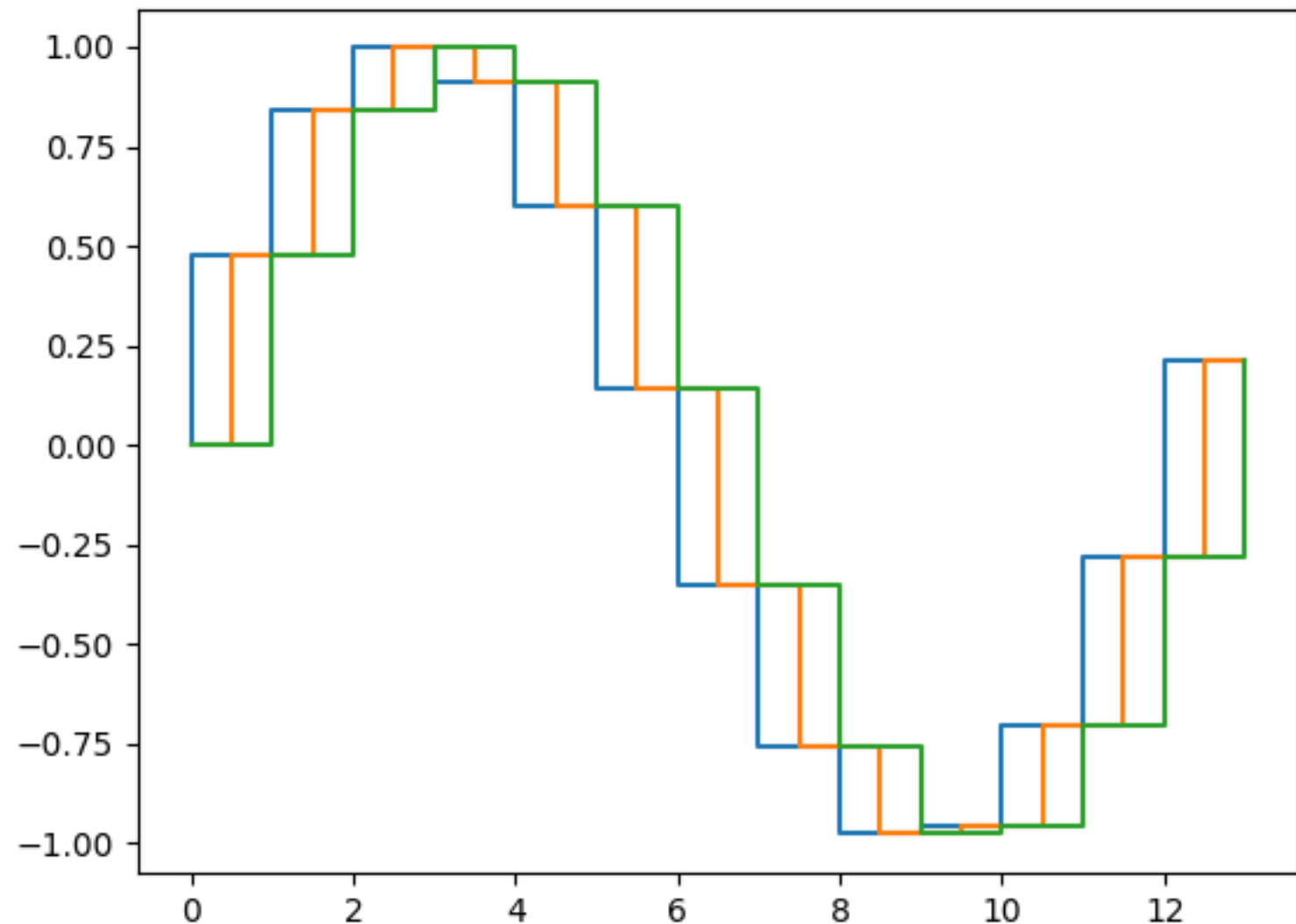


Matplotlib (step)

- Desenhar linhas retas em vez de curvas
- Podemos controlar a posição de cada ponto em relação ao nível

```
x = np.arange(14)
y = np.sin(x / 2)

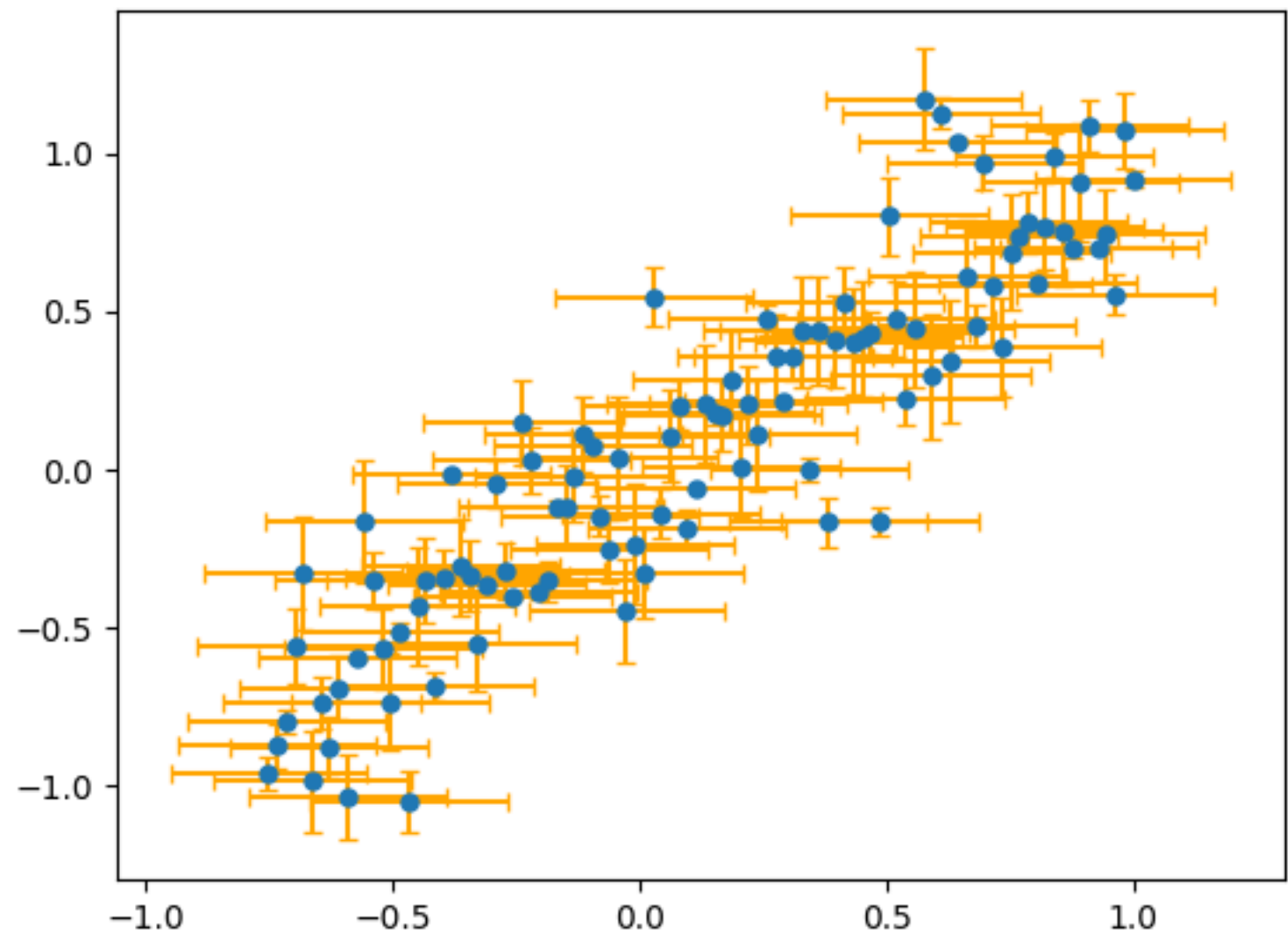
plt.step(x, y) # blue
plt.step(x, y,
where='mid') # orange
plt.step(x, y,
where='post') # green
plt.show()
```



Matplotlib (errorbar)

- Atribuir uma margem de erro a cada dimensão
- Podemos controlar desenho dos intervalos

```
xx = np.linspace(-0.75, 1., 100)
yy = xx +
0.25*np.random.randn(len(xx))
sigma_y = np.random.uniform(0, 0.2,
len(xx))
plt.errorbar(xx, yy, yerr=sigma_y,
xerr=0.2, fmt='o', markersize=5,
ecolor="orange", capsize=3)
plt.show()
```

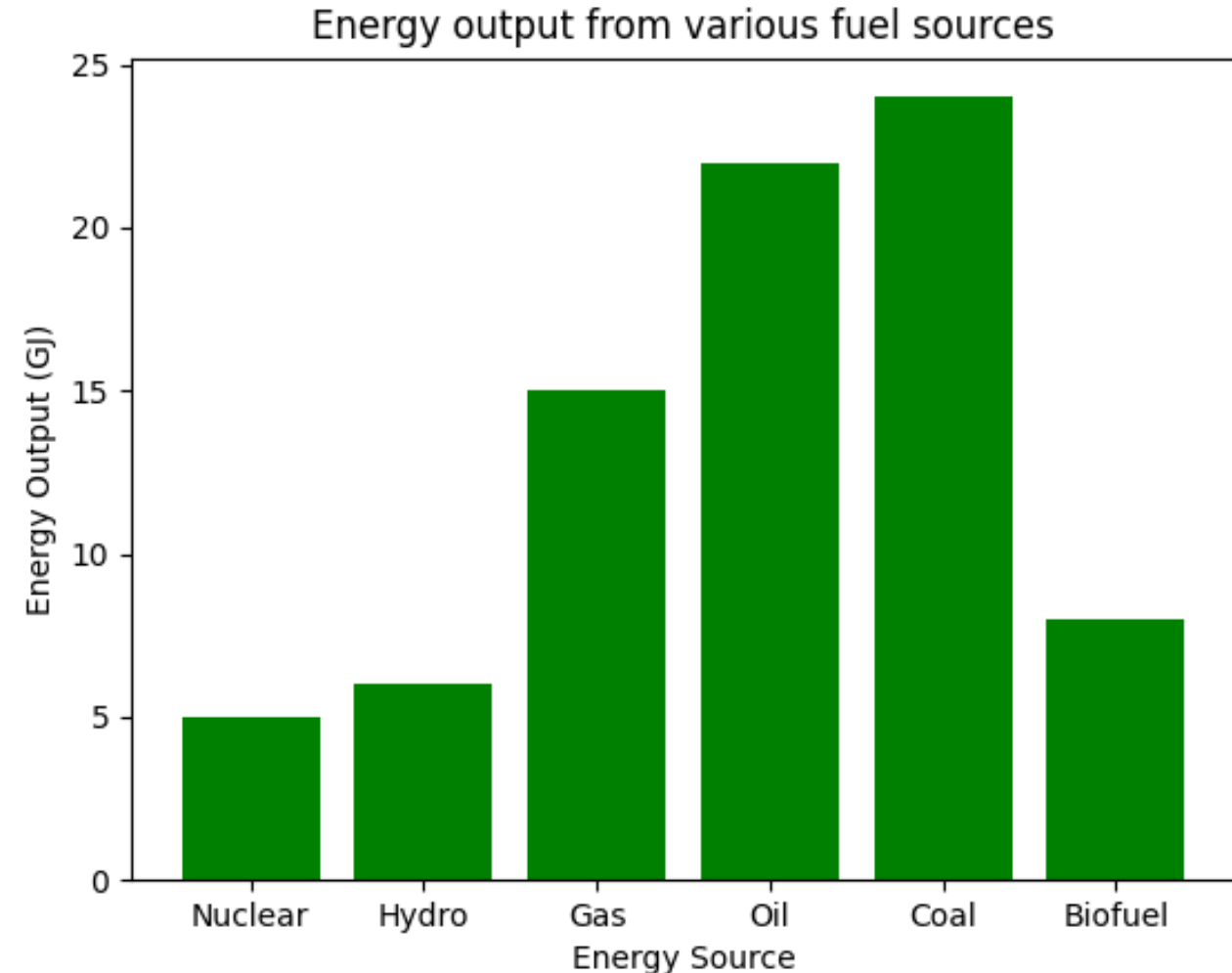


Matplotlib (bar)

```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, energy, color='green')
plt.xlabel("Energy Source")
plt.ylabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")
plt.xticks(x_pos, x)
plt.show()
```

- Desenhar cada ponto como uma barra vertical

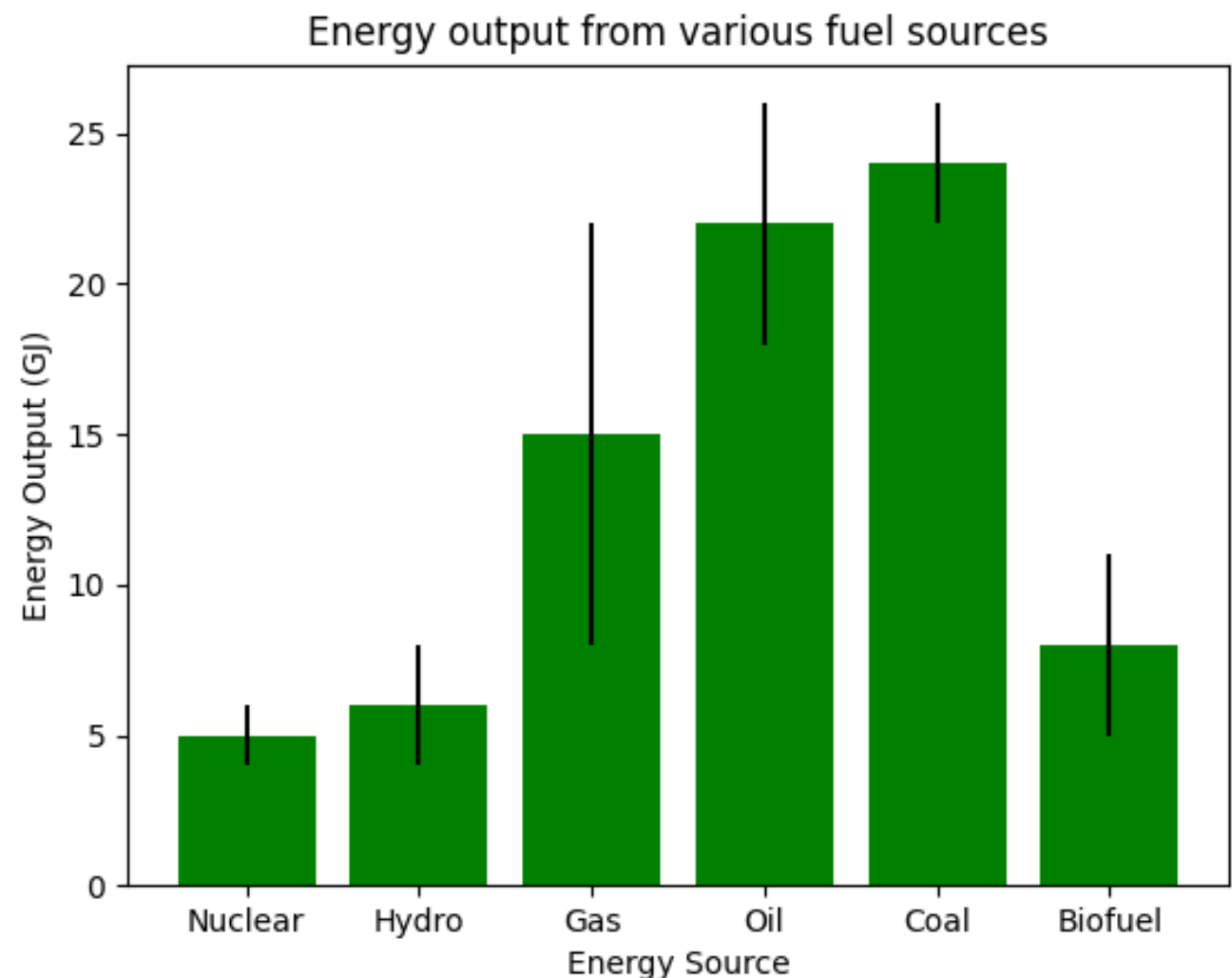


Matplotlib (bar)

```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]
x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, energy, color='green', yerr=variance)
plt.xlabel("Energy Source")
plt.ylabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")
plt.xticks(x_pos, x)
plt.show()
```

- Podemos acrescentar uma margem de erro

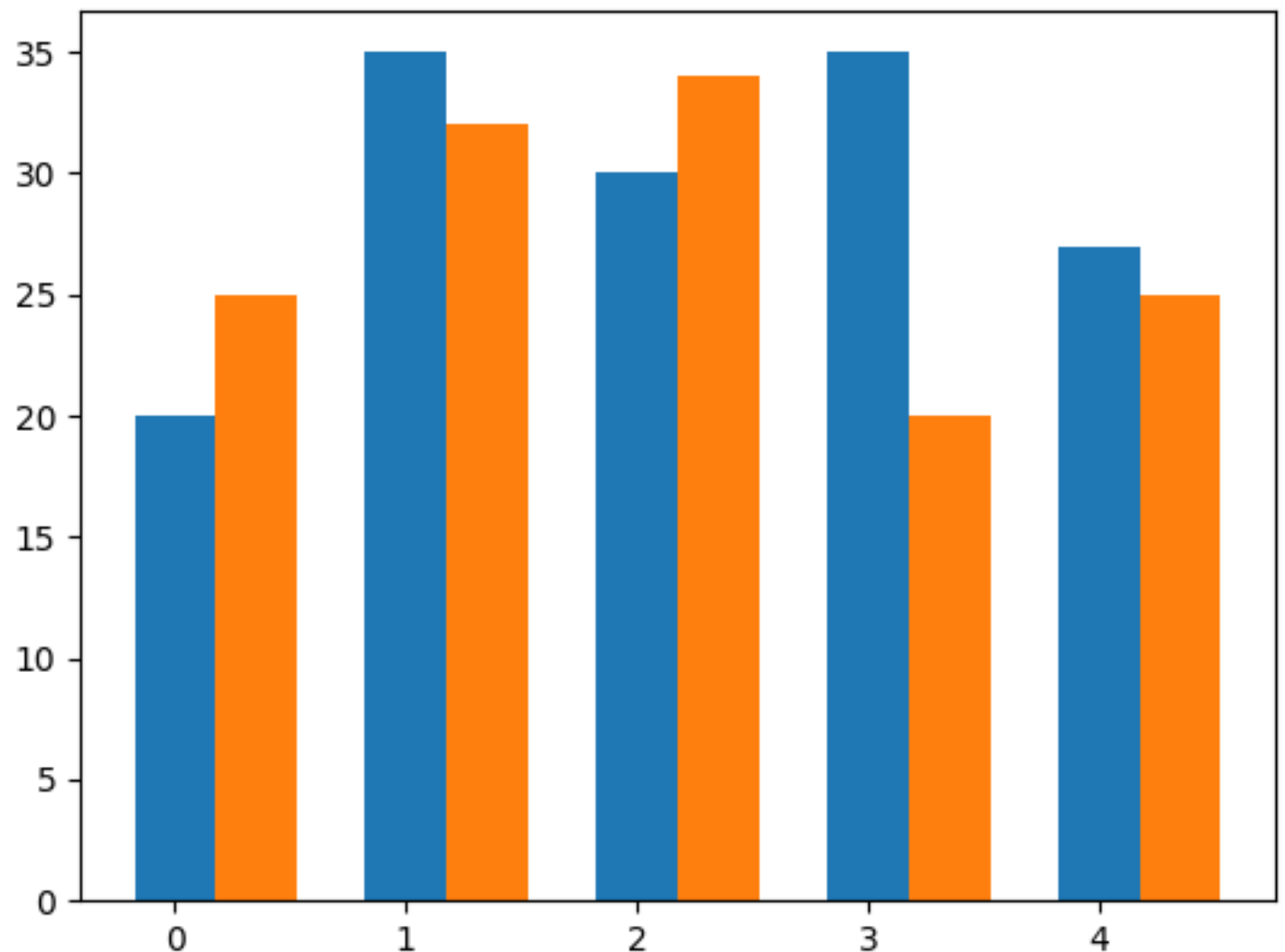


Matplotlib (bar)

```
xx = np.arange(5)
men_means = (20, 35, 30, 35, 27)
women_means = (25, 32, 34, 20, 25)

width=0.35
plt.bar(xx,men_means,width=width)
plt.bar(xx + width,women_means,width=width)
plt.show()
```

- Podemos desenhar múltiplas barras
- Controlando a posição nos X

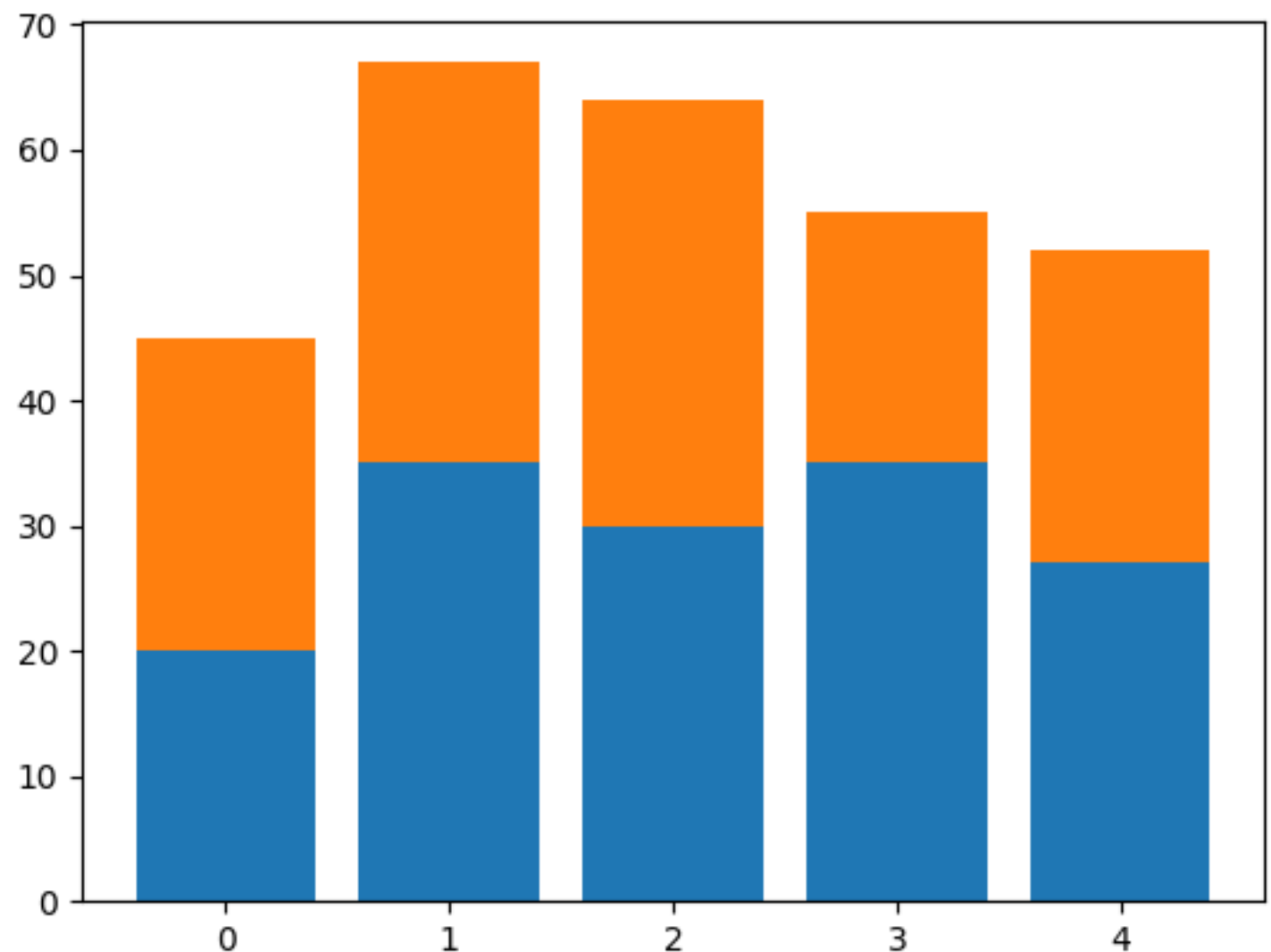


Matplotlib (bar)

```
xx = np.arange(5)
men_means = (20, 35, 30, 35, 27)
women_means = (25, 32, 34, 20, 25)

plt.bar(xx, men_means)
plt.bar(xx, women_means, bottom=men_means)
plt.show()
```

- Podemos desenhar múltiplas barras
- Controlando a posição nos Y



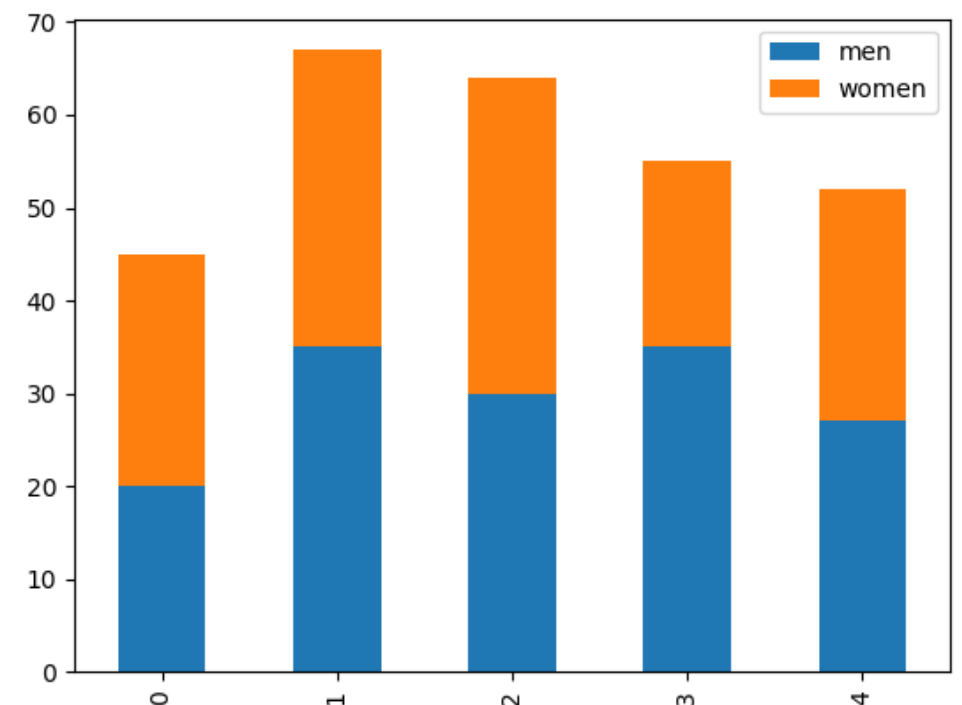
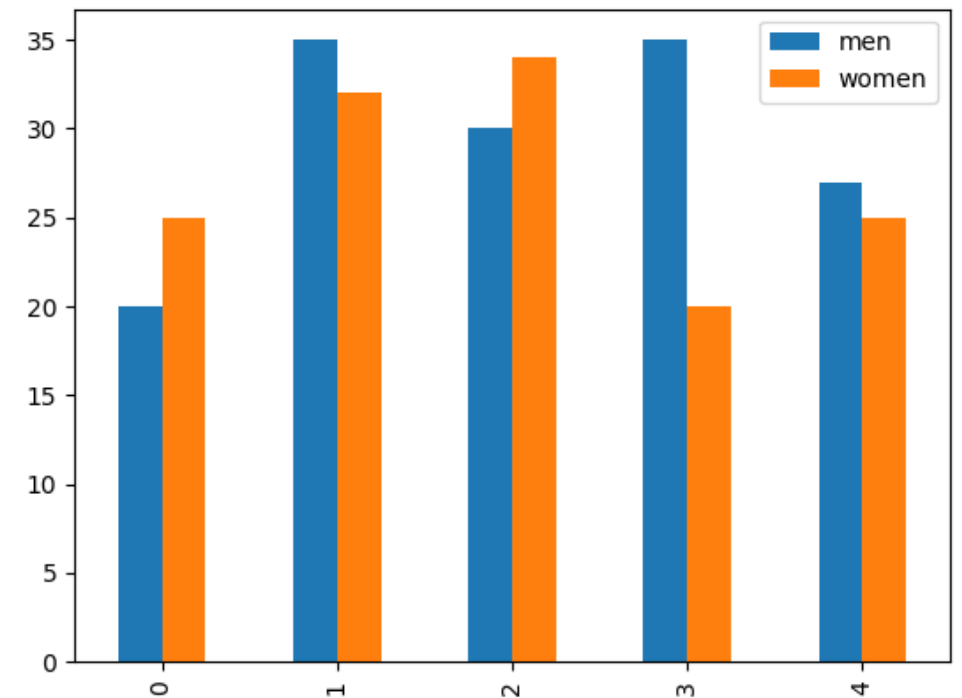
Matplotlib (bar)

- Desenhar múltiplas barras é mais simples utilizando um *DataFrame*

```
men_means = (20, 35, 30, 35, 27)
women_means = (25, 32, 34, 20, 25)
df =
pd.DataFrame({'men':men_means, 'women':women_means})
```

```
# lado a lado
df.plot(kind='bar')
plt.show()
```

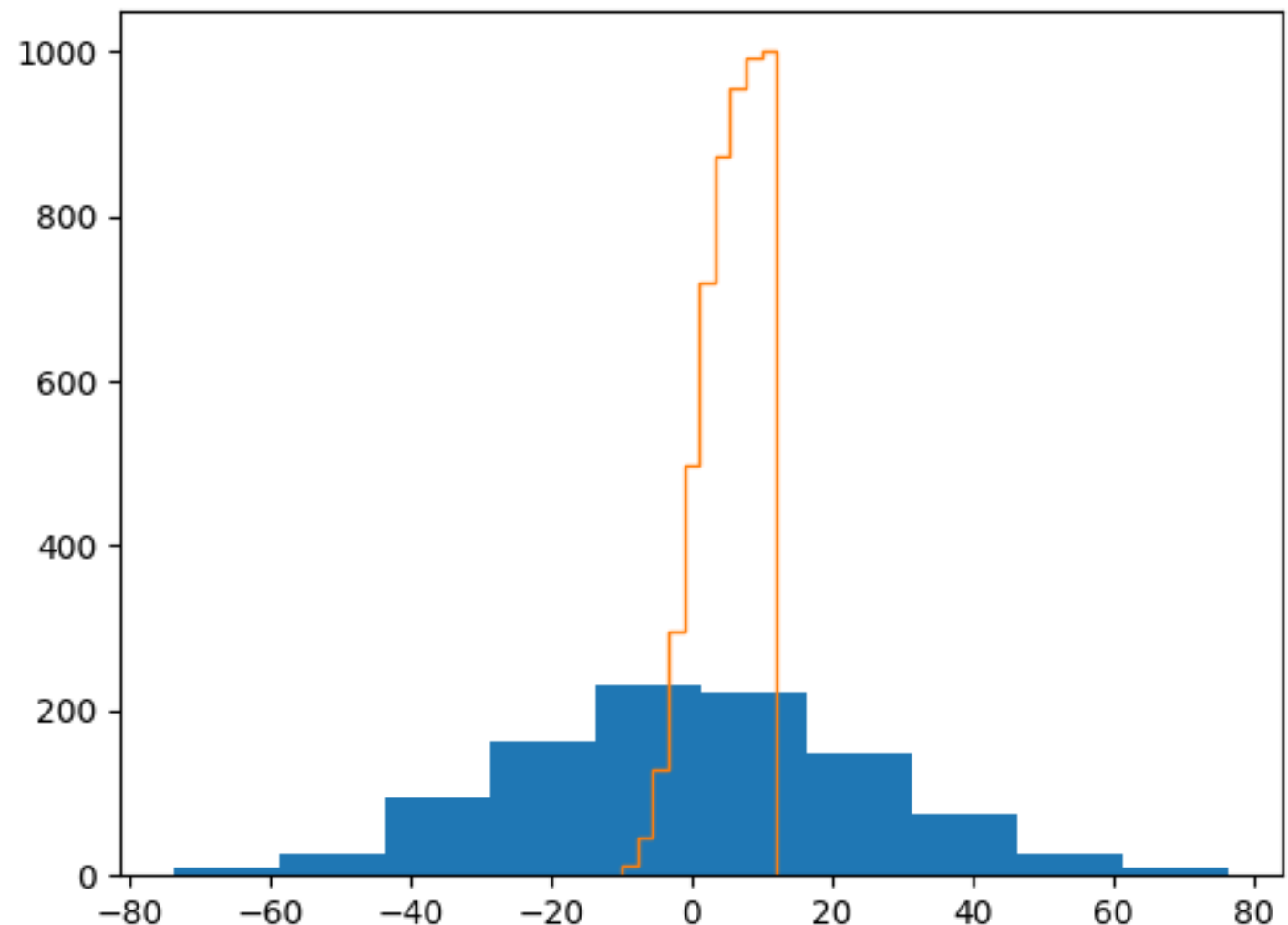
```
# em cima uma da outra
df.plot(kind='bar', stacked=True)
plt.show()
```



Matplotlib (hist)

- Desenhar um histograma a partir de uma sequência
 - X = valores na sequência
 - Y = número de ocorrências de cada valor

```
data1 = np.random.normal(1.,  
25., 1000)  
data2 = np.random.normal(1.,  
4., 1000)  
plt.hist(data1)  
plt.hist(data2, histtype="step",  
cumulative=True)  
plt.show()
```

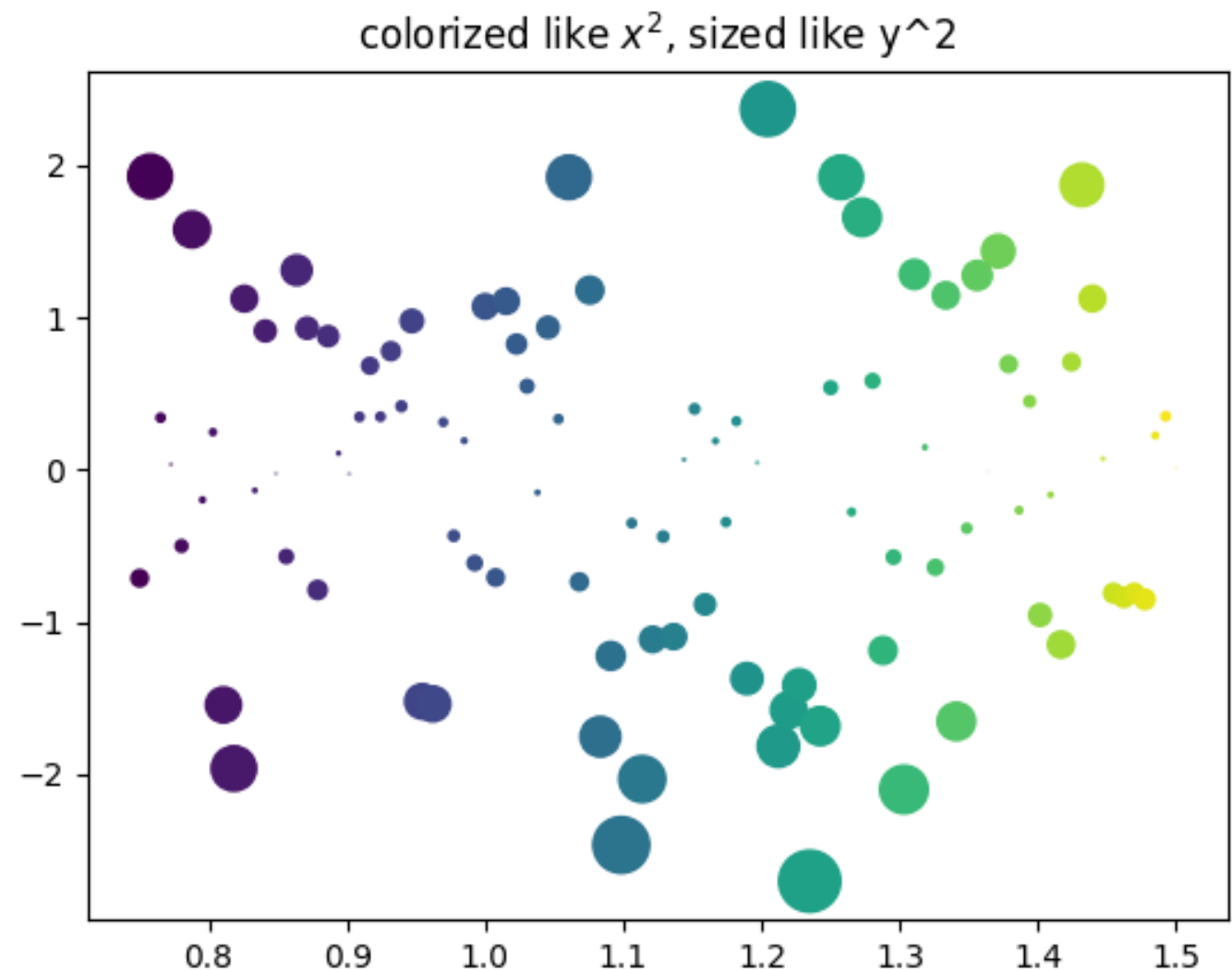


Matplotlib (scatter)

- Desenhar cada ponto independentemente
- Podemos controlar cor e tamanho de cada ponto

```
xx = np.linspace(0.75, 1.5, 100)
yy = np.random.randn(len(xx))
zz = xx**2
ww = yy**2

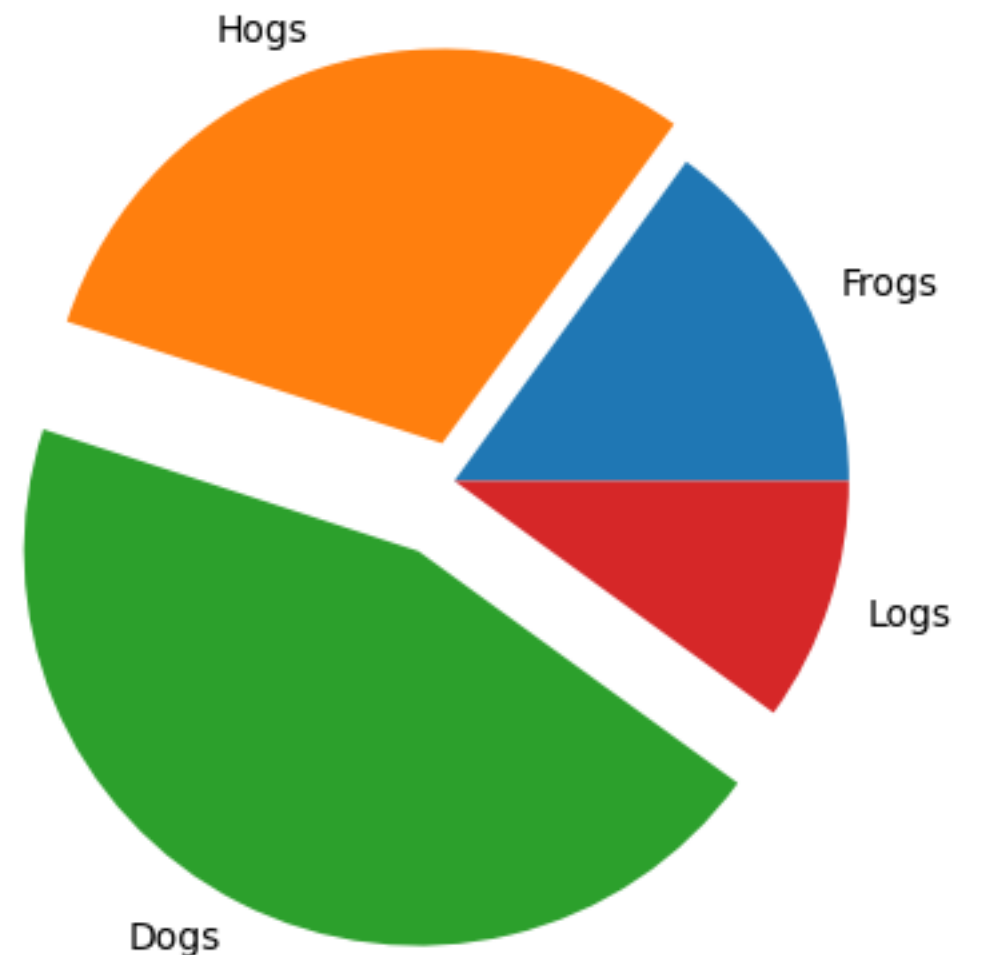
plt.scatter(xx, yy, c=zz, s=ww*50)
plt.title("colorized like  $x^2$ ,  
sized like  $y^2$ ")
plt.show()
```



Matplotlib (pie)

- Desenhar um gráfico em forma de “tarte”
- Podemos controlar expansão de cada fatia

```
labels = ['Frogs', 'Hogs', 'Dogs',  
          'Logs']  
sizes = [15, 30, 45, 10]  
explode = (0, 0.1, 0.2, 0)  
plt.pie(sizes, explode=explode,  
        labels=labels, autopct=lambda x:x)  
plt.show()
```



Exemplo 1

- Desenhar um gráfico de áreas que apresente a percentagem de testes COVID-19 de cada tipo ao longo do tempo

```
amostras = pd.read_csv('amostras.csv')  
amostras.dropna(inplace=True)
```

```
amostras['pcr'] = amostras['amostras_pcr'] / amostras['amostras']  
amostras['antigenio'] = amostras['amostras_antigenio'] /  
amostras['amostras']  
amostras[['pcr', 'antigenio']].plot(kind='area', stacked=True)
```

```
plt.tick_params(axis='x', bottom=False, labelbottom=False)  
plt.ylim(0.4, 1)  
plt.yticks([0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],  
["40%", "50%", "60%", "70%", "80%", "90%", "100%"])  
plt.show()
```



Exemplo 2

- Dados climatéricos anuais de longa duração fornecidos pelo IPMA [aqui](#).
- Gráfico da evolução das temperaturas mínima e máxima por ano

```
dfs = pd.read_excel("PT100-tx-tn-prec.xlsx", sheet_name=None)
years = ['year', 'Annual']
```

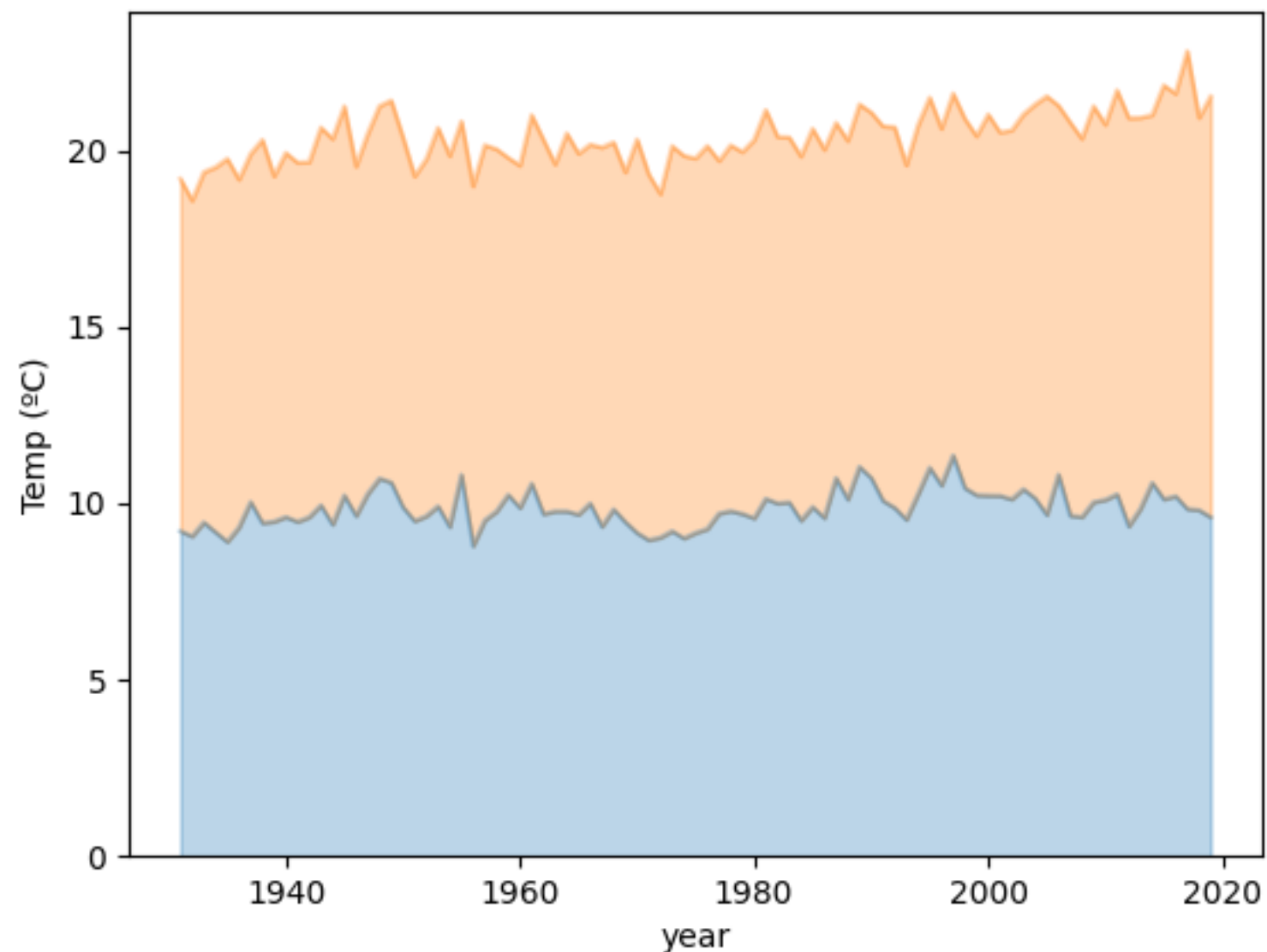
```
tmins = dfs['tmin']
tmins = tmins[years].copy()
tmins.dropna(inplace=True)
```

```
tmaxs = dfs['tmax']
tmaxs = tmaxs[years].copy()
tmaxs.dropna(inplace=True)
```

```
temps = pd.merge(tmins, tmaxs, on='year', suffixes=('_tmin', '_tmax'))
temps['year'] = temps['year'].astype('uint16')
temps.set_index('year', inplace=True)
temps['Annual_tmax'] = temps['Annual_tmax'] - temps['Annual_tmin']
temps.plot(kind='area', alpha=0.3, stacked=True, legend=False, ylabel=
'Temp (°C)')
```


Exemplo 2

- Dados climatéricos anuais de longa duração fornecidos pelo IPMA [aqui](#).
- Gráfico da evolução das temperaturas mínima e máxima por ano



Exemplo 3

- Dados climatéricos anuais de longa duração fornecidos pelo IPMA [aqui](#).
- Gráfico de ano, temperatura mínima e temperatura máxima por estação
- E.g., qual o Verão mais quente de que há registo?

```
dfs = pd.read_excel("PT100-tx-tn-prec.xlsx", sheet_name=None)
```

```
seasons = ['Spring', 'Summer', 'Autumn', 'Winter']  
yseasons = ['year'] + seasons
```

```
# temperaturas mínimas para cada ano e estação  
tmins = dfs['tmin']  
tmins = tmins[yseasons].copy()  
tmins.dropna(inplace=True)  
tmins['year'] = tmins['year'].astype('uint16')
```

```
# temperaturas máximas para cada ano e estação  
tmaxs = dfs['tmax']  
tmaxs = tmaxs[yseasons].copy()  
tmaxs.dropna(inplace=True)  
tmaxs['year'] = tmaxs['year'].astype('uint16')
```

Exemplo 3

- Calcular temperaturas mínimas e máximas por estação, e anos correspondentes

```
# colapsar as estações para índices
tmins = tmins.melt(id_vars=["year"], var_name="season", value_name="tmin")
tmins.set_index('year', inplace=True)
years = tmins.groupby('season').idxmin().rename(columns={'tmin': 'year'})
temps = tmins.groupby('season').min()
tmins = years.join(temps)
```

```
# colapsar as estações para índices
tmaxs = tmaxs.melt(id_vars=["year"], var_name="season", value_name="tmax")
tmaxs.set_index('year', inplace=True)
years = tmaxs.groupby('season').idxmax().rename(columns={'tmax': 'year'})
temps = tmaxs.groupby('season').max()
tmaxs = years.join(temps)
```

Exemplo 3

```
for season in seasons:
    plt.scatter(tmins['year'][season], tmins['tmin'][
season], c=season_color[season], s=200, marker="v")
    plt.scatter(tmaxs['year'][season], tmaxs['tmax'][season],
c=season_color[season], s=200, marker="^")
    plt.scatter([], [], c=season_color[season], s=100, marker='s', label=season)
plt.legend()
plt.show()
```

- Desenhar o gráfico

