Métodos Quantitativos
Aula 01

Visualização de Dados

Roberto Massi de Oliveira
Alex Borges Vieira

Visualização de Dados

- Essencial em *Data Science*, principalmente, por possibilitar:
  - **Análise exploratória de dados:** como as amostras coletadas se comportam? A visualização gráfica pode dar as primeiras orientações sobre como trabalhar com um conjunto de dados
  - **Detecção de erros:** percepção de outliers, de limpezas insuficientes nos dados e de pressupostos inadequados
  - **Comunicação:** você é capaz de apresentar seus resultados para terceiros de forma eficaz? Resultados significativos só fazem sentido se compartilhados. Seu sucesso depende da sua capacidade de convencer o próximo que seus resultados são bons

Análise Exploratória de Dados

- O método científico tradicional é orientado por hipóteses que antecedem a manipulação dos dados
  - O pesquisador formula uma hipótese de como o mundo funciona e, em seguida, procura apoiar ou rejeitar essa hipótese com base em dados
- A proposta dessa disciplina, ciência orientada a dados, difere do método tradicional, com a manipulação dos dados antecedendo as hipóteses
  - Começaremos reunindo um conjunto de dados substancial e, em seguida, procuraremos padrões que, idealmente, desempenharão o papel de hipóteses para análises futuras
- Análise exploratória de dados é a busca de padrões e tendências em um determinado conjunto de dados
  - As técnicas de visualização desempenham um papel importante nessa busca

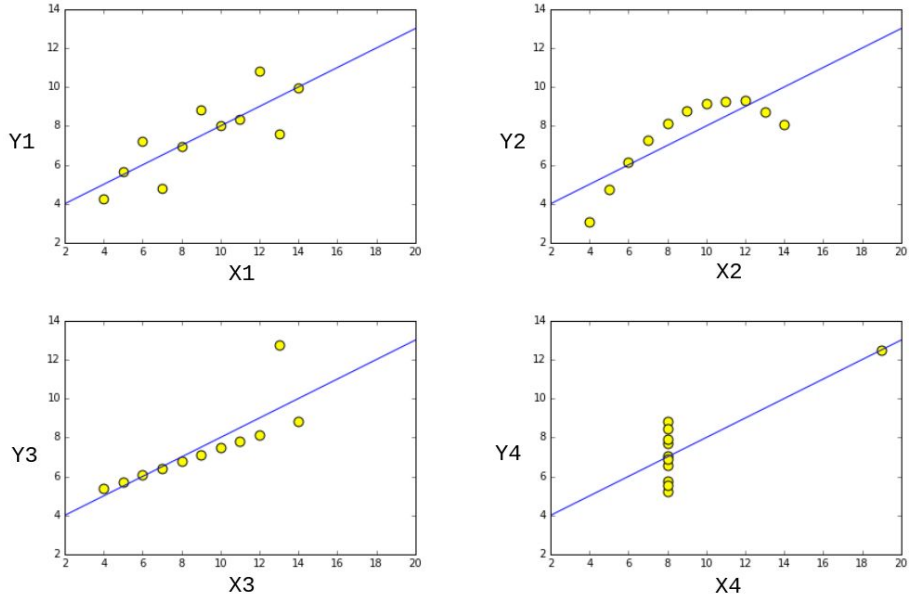
Análise Exploratória de Dados

- Existe um limite do quanto se pode entender os dados sem gráficos
- Ex. (Quarteto de Anscombe):

	I		II		III		IV	
	x	y	x	y	x	y	x	y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.31	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Mean	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Var.	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Corr.	0.816		0.816		0.816		0.816	

Análise Exploratória de Dados

- Ex. (Quarteto de Anscombe):
  - Análise estatística sumarizada na tabela nos leva a crer que as 4 amostras são similares
  - Mas e se traçarmos gráficos? Essa ideia é mantida?



Análise Exploratória de Dados

- Ao confrontar novos conjuntos de dados, o que fazer?
  1. Responder questões como:
     a. Quem gerou esse conjunto de dados, quando e por que?
     b. Quão grande é esse conjunto de dados? Quão variadas são suas informações?
     c. Qual o significado de cada campo do conjunto de dados?
  2. Procurar registros familiares ou interpretáveis

Age
Weight
Height
Leg Length
Arm Length
Arm Circumference
Waist

Análise Exploratória de Dados

- Ao confrontar novos conjuntos de dados, o que fazer (continuação)?

3. Extrair e observar estatísticas como médias, desvios padrões, máximos, mínimos, etc.

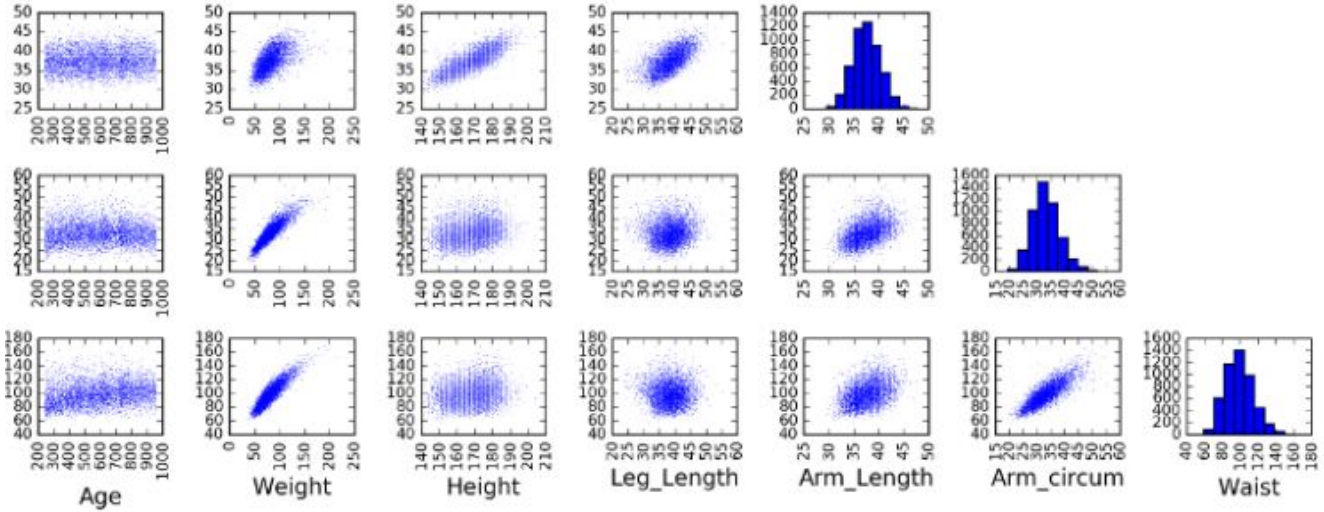
	Min	25%	Median	75%	Max
Age	241	418	584	748	959
Weight	32.4	67.2	78.8	92.6	218.2
Height	140	160	167	175	204
Leg Length	23.7	35.7	38.4	41	55.5
Arm Length	29.5	35.5	37.4	39.4	47.7
Arm Circumference	19.5	29.7	32.8	36.1	141.1
Waist	59.1	87.5	97.95	108.3	172

4. Investigar correlações, extrair coeficientes de correlação, matriz de correlação

	Age	Weight	Height	Leg Length	Arm Length	Arm Circum	Waist
Age	1.000						
Weight	0.017	1.000					
Height	−0.105	0.443	1.000				
Leg_Len	−0.268	0.238	0.745	1.000			
Arm_Len	0.053	0.583	0.801	0.614	1.000		
Arm_Circ	0.007	0.890	0.226	0.088	0.444	1.000	
Waist	0.227	0.892	0.181	−0.029	0.402	0.820	1.000

Análise Exploratória de Dados

- Ao confrontar novos conjuntos de dados, o que fazer (continuação)?
  5. Tem como classificar, agrupar dados da amostra? Sumarizar por gênero, localização, etc.?
  6. Plotagens das amostras e das distribuições estatísticas



Bibliotecas

- Daremos enfoque no que está em negrito:
  - Estatística:
    - NumPy
    - SciPy
    - **Pandas**
    - StatsModels
  - Visualização:
    - **Matplotlib** (matplotlib.pyplot)
    - Seaborn
    - Plotly
    - Bokeh
- Neste ponto, é muito importante o uso do Jupyter Notebook

Bibliotecas

- Pandas
  - Fundamental na manipulação de tabelas/estruturas de dados
  - Em ambientes notebook, fica fácil a visualização de tais estruturas como um ***data frame***
  - Pode ler dados de formatos como .xls, por exemplo, e interpretar no ambiente Python

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        %matplotlib inline
```

```
In [2]: titanic_df = pd.read_excel('titanic3.xls','titanic3', index_col=None, na_values=['NA']) #read excel sheet to dataframe
```

```
In [3]: titanic_df.head()
```

Out[3]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON

Bibliotecas

- Pandas
  - Um único comando é capaz de extrair inúmeras estatísticas da amostra coletada

```
In [4]: titanic_df.describe()
```

Out[4]:

	pclass	survived	age	sibsp	parch	fare	body
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000	121.000000
mean	2.294882	0.381971	29.881135	0.498854	0.385027	33.295479	160.809917
std	0.837836	0.486055	14.413500	1.041658	0.865560	51.758668	97.696922
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800	72.000000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200	155.000000
75%	3.000000	1.000000	39.000000	1.000000	0.000000	31.275000	256.000000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	328.000000

Bibliotecas

- Pandas
  - A limpeza dos dados também é realizada de forma simples

```
titanic_df.drop(['ticket','cabin','boat','body'],axis=1).head()
```

	pclass	survived	name	sex	age	sibsp	parch	fare	embarked	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	211.3375	S	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	151.5500	S	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	151.5500	S	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	151.5500	S	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	151.5500	S	Montreal, PQ / Chesterville, ON

Bibliotecas

- Pandas
  - Também é possível fazer agrupamentos

```
In [11]: titanic_df.groupby(['sex']).mean()
```

Out[11]:

	pclass	survived	age	sibsp
sex				
female	2.154506	0.727468	28.687071	0.652361
male	2.372479	0.190985	30.585233	0.413998

```
In [13]: titanic_df[titanic_df['age']<18].groupby(['sex','pclass']).mean()
```

Out[13]:

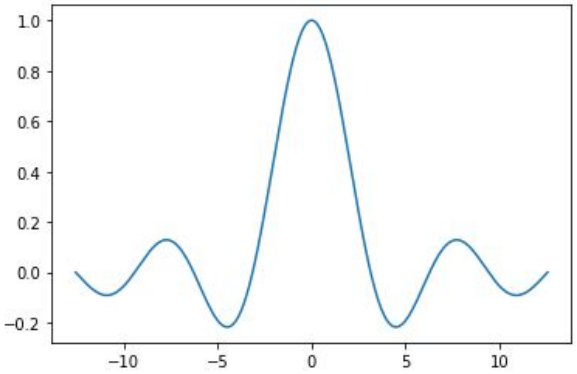
		survived	age	sibsp	parch	fare	body
sex	**pclass**						
female	**1**	0.875000	14.125000	0.500000	0.875000	104.083337	NaN
	2	1.000000	8.273150	0.666667	1.166667	27.998844	NaN
	3	0.543478	8.416667	1.456522	1.043478	18.284148	328.0
male	**1**	0.857143	9.845243	0.571429	1.714286	129.752371	NaN
	2	0.733333	6.232220	0.600000	0.933333	31.750280	NaN
	3	0.233333	9.838888	1.966667	1.016667	21.677570	65.5

- Pesquisem mais funcionalidades e façam testes!

Bibliotecas

- Matplotlib

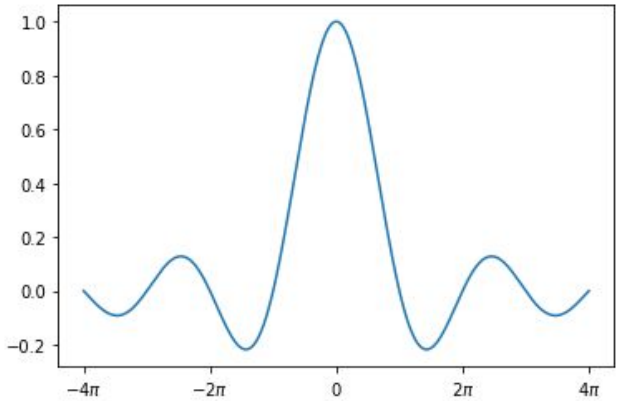
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 pi = np.pi
5 x = np.linspace(-4*pi, 4*pi, 1000)
6
7 plt.plot(x, np.sin(x)/x)
8 plt.show()
```



Bibliotecas

- Matplotlib
  - Observem o comando plt.xticks e o efeito do mesmo no eixo X do gráfico

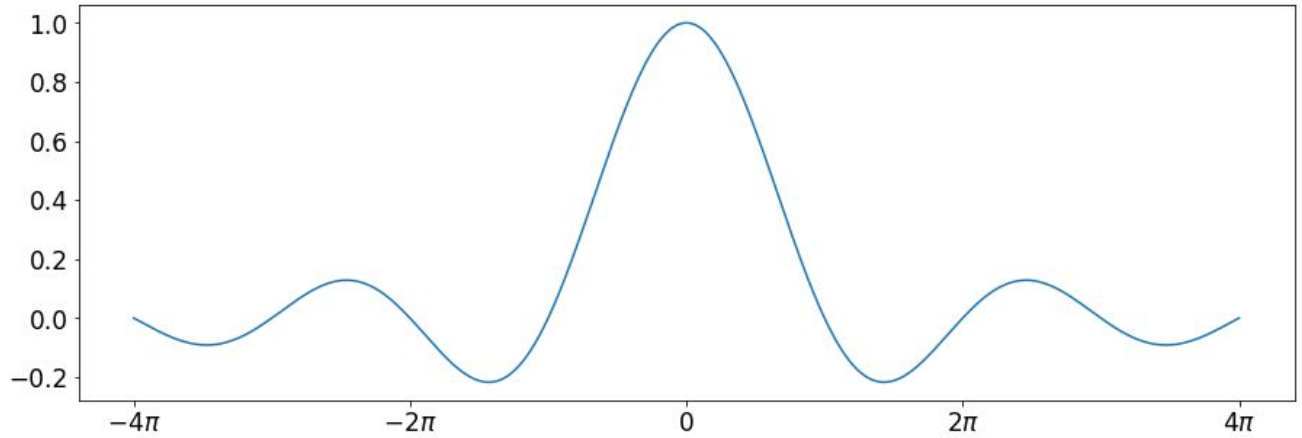
```
7  plt.xticks([-4*pi, -2*pi, 0, 2*pi, 4*pi],
8             ['$-4\pi$','$-2\pi$','$0$','$2\pi$','$4\pi$'])
9
10 plt.plot(x, np.sin(x)/x)
11 plt.show()
```



Bibliotecas

- Matplotlib
  - Observem os comando plt.rcParams e o efeito do mesmo no tamanho do gráfico e da fonte

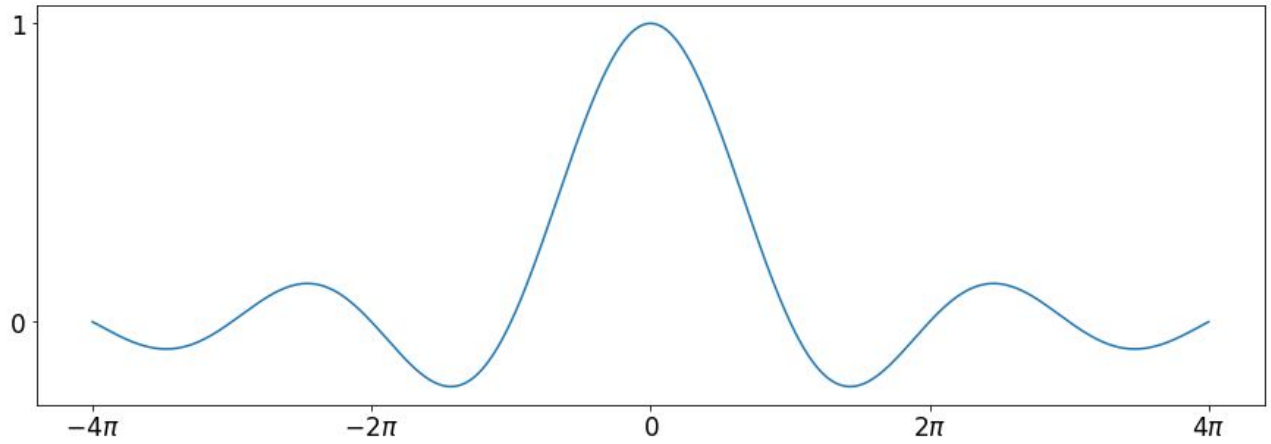
```
7  plt.rcParams['figure.figsize'] = (15,5)
8  plt.rcParams.update({'font.size': 17})
9
10 plt.xticks([-4*pi, -2*pi, 0, 2*pi, 4*pi],
11           ['$-4\pi$','$-2\pi$','$0$','$2\pi$','$4\pi$'])
12
13 plt.plot(x, np.sin(x)/x)
14 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos arrumar agora os yticks

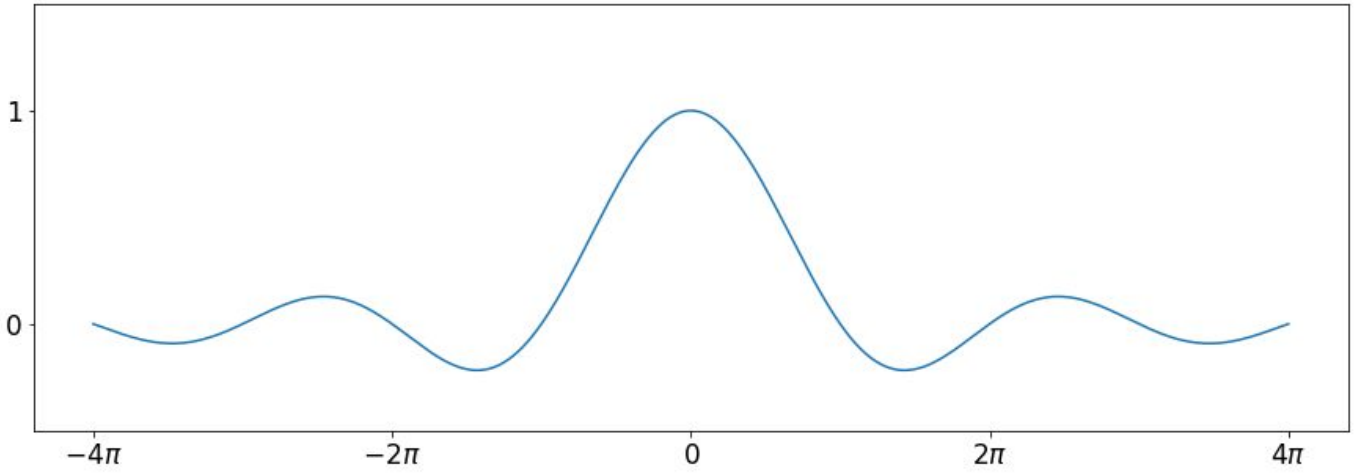
```
10 plt.xticks([-4*pi, -2*pi, 0, 2*pi, 4*pi],
11           ['$-4\pi$','$-2\pi$','$0$','$2\pi$','$4\pi$'])
12
13 plt.yticks([0,1],['$0$','$1$'])
14
15 plt.plot(x, np.sin(x)/x)
16 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos centralizar melhor o gráfico mexendo nos limites do eixo y (plt.ylim)

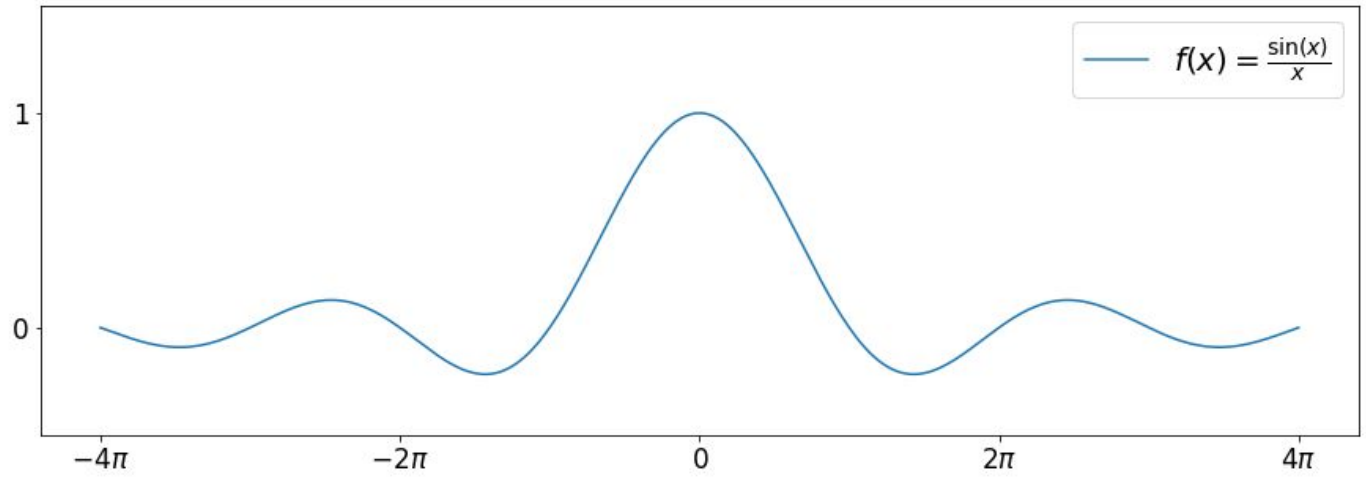
```
15 plt.ylim(-0.5, 1.5)
16
17 plt.plot(x, np.sin(x)/x)
18 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos colocar legenda especificando um label e um plt.legend

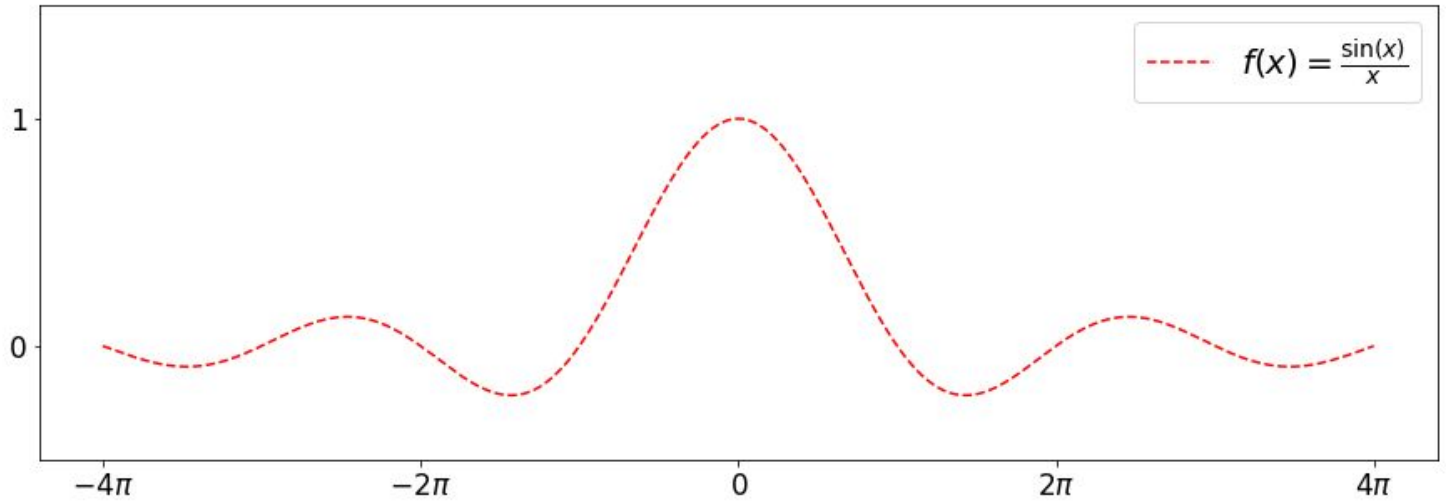
```
17 plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$')
18 plt.legend(loc='best',fontsize=20)
19
20 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos modificar a cor e o estilo da linha (parâmetros color e linestyle)

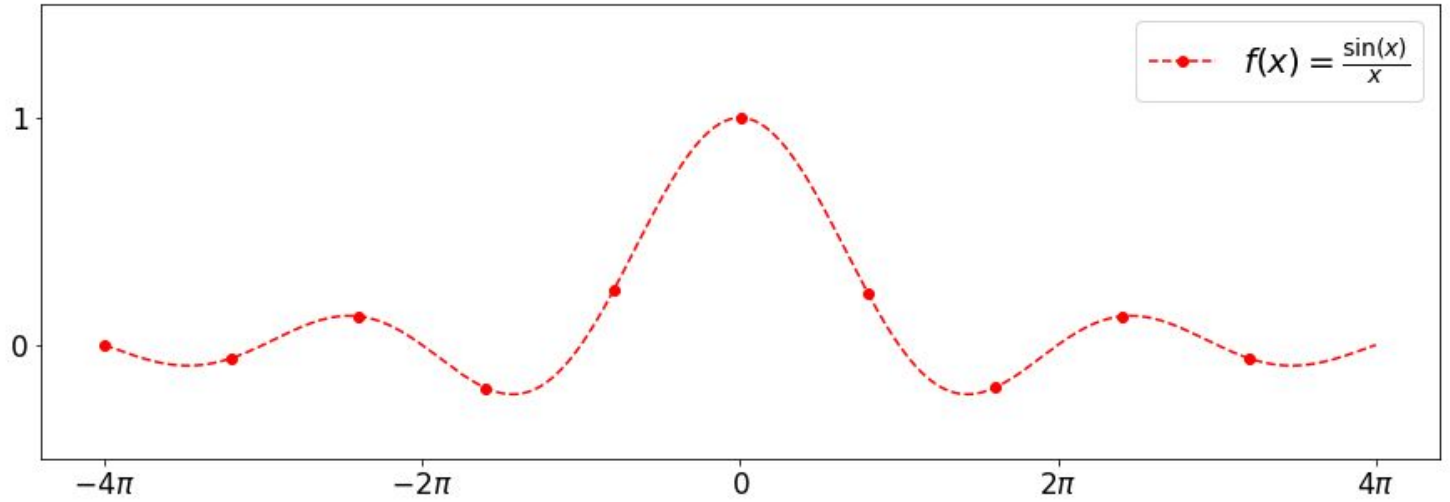
```
18 plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$', color='r', linestyle='--')
19 plt.legend(loc='best',fontsize=20)
20
21 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos colocar marcadores (marker) espaçados a cada 100 pontos da curva (markevery)

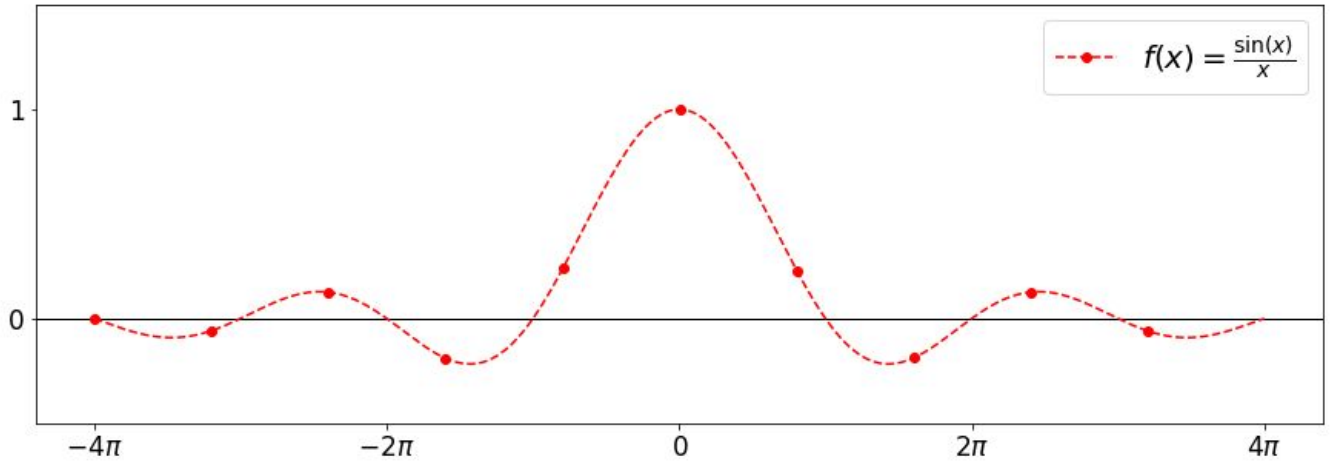
```
18 plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$', color='r', linestyle='--', marker='o', markevery=100)
19 plt.legend(loc='best',fontsize=20)
20
21 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos colocar uma linha passando pela origem (plt.axhline), para facilitar o estudo da curva

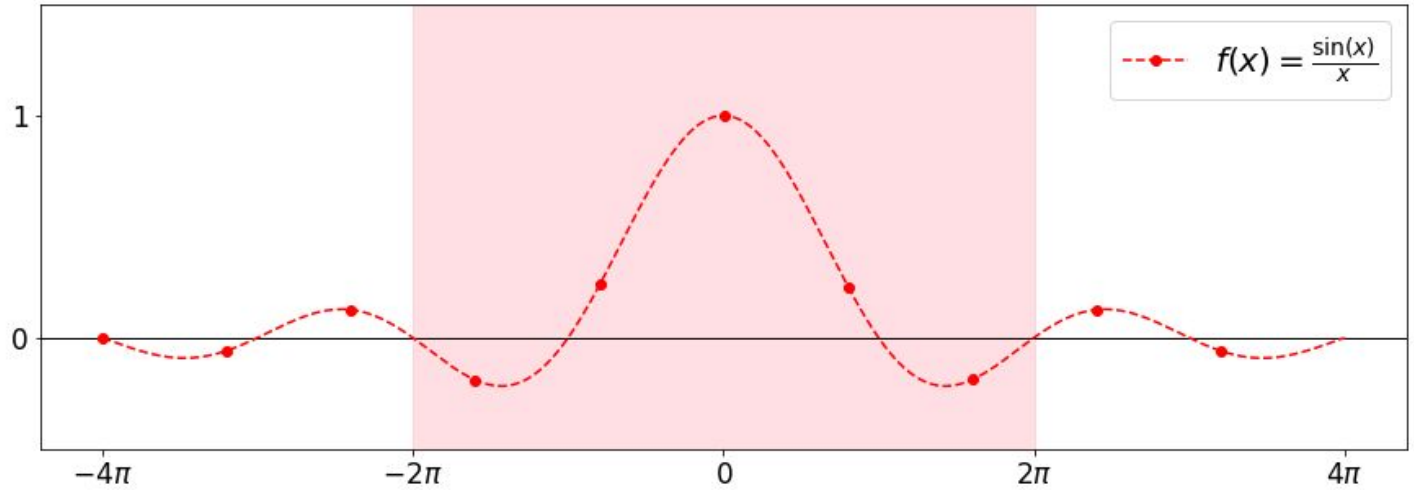
```
16 plt.axhline(0,color='black',lw=1)
17
18 plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$', color='r', linestyle='--', marker='o', markevery=100)
19 plt.legend(loc='best',fontsize=20)
20
21 plt.show()
```



Bibliotecas

- Matplotlib
  - Vamos destacar uma área específica do gráfico (plt.axvspan)

```
18 plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$', color='r', linestyle='--', marker='o', markevery=100)
19 plt.axvspan(-2*pi, 2*pi, color='pink', alpha=0.5)
20 plt.legend(loc='best',fontsize=20)
21
22 plt.show()
```



Bibliotecas

- Matplotlib
  - Vejamos o código completo (https://drive.google.com/open?id=1_ccRZZ71R0mwtJpO5MgAZ3pRa3xx19m5)

```python
import numpy as np
import matplotlib.pyplot as plt

pi = np.pi
x = np.linspace(-4*pi, 4*pi, 1000)

plt.rcParams['figure.figsize'] = (15,5)
plt.rcParams.update({'font.size': 17})

plt.xticks([-4*pi, -2*pi, 0, 2*pi, 4*pi],['$-4\pi$','$-2\pi$','$0$','$2\pi$','$4\pi$'])
plt.yticks([0,1],['$0$','$1$'])
plt.ylim(-0.5, 1.5)

plt.axhline(0,color='black',lw=1)

plt.plot(x, np.sin(x)/x, label=r'$f(x)=\frac{\sin(x)}{x}$', color='r', linestyle='--', marker='o', markevery=100)
plt.axvspan(-2*pi, 2*pi, color='pink', alpha=0.5)
plt.legend(loc='best',fontsize=20)

plt.show()
```

  - Pesquisem mais funcionalidades e façam testes!
    (https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

Bibliotecas

- Ex. (Pandas+Matplotlib):
  - Importando dados para um *data frame* do Pandas

```
In [1]: import pandas as pd
        dados = pd.read_csv('C:/Users/Natanael/Documents/DidaticaTech/athlete_events.csv')
```

```
In [2]: dados.head()
```

Out[2]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

Bibliotecas

- Ex. (Pandas+Matplotlib):
  - Separando apenas pessoas do sexo masculino em uma nova variável (novo *data frame*)

```
In [3]: masculinos = dados.loc[dados['Sex']=='M']
```

```
In [4]: masculinos.head()
```

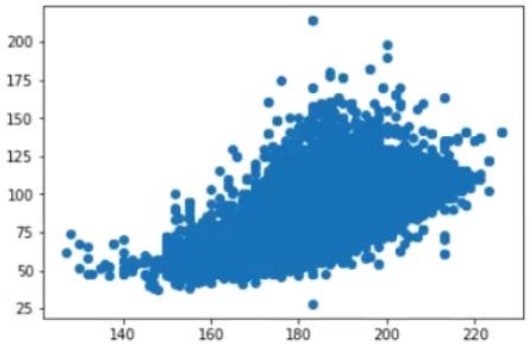
Out[4]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
10	6	Per Knut Aaland	M	31.0	188.0	75.0	United States	USA	1992 Winter	1992	Winter	Albertville	Cross Country Skiing	Cross Country Skiing Men's 10 kilometres	NaN

Bibliotecas

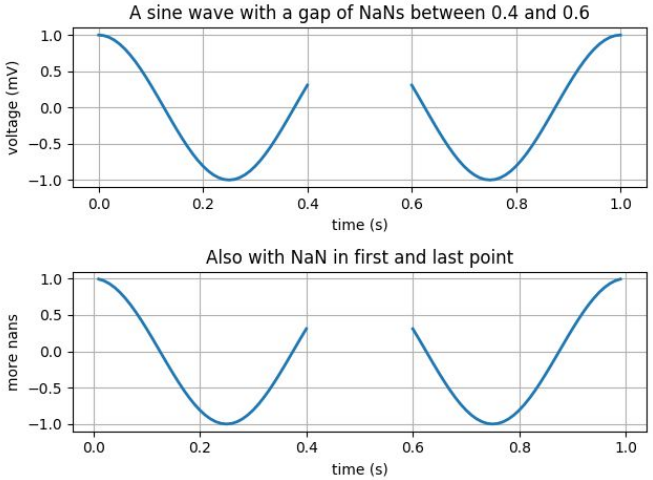
- Ex. (Pandas+Matplotlib):
  - Plotando gráfico de dispersão de peso pela altura. O que se pode concluir a priori?

```
In [3]: masculinos = dados.loc[dados['Sex']=='M']

In [5]: a = masculinos['Height']
        p = masculinos['Weight']

In [6]: import matplotlib.pyplot as plt
        plt.scatter(a,p)
        plt.show()
```



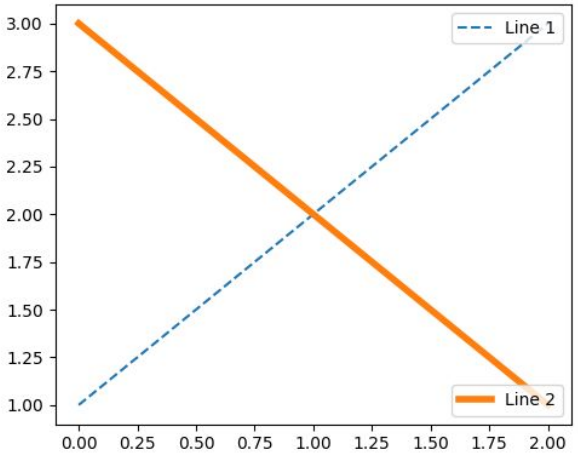
Dicas

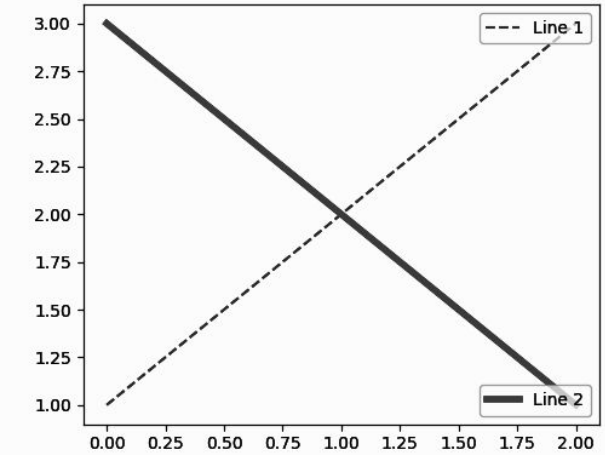
- Deixar todos os gráficos referentes a uma dada métrica na mesma escala
  - xlim, xticks, ylim, yticks, escalas logarítmicas ou não, etc
- Achatar gráficos com o comando rcParams[‘figure.figsize’] (ganhar espaço)
- O comando grid() pode ser importante para facilitar as análises dos gráficos



Dicas

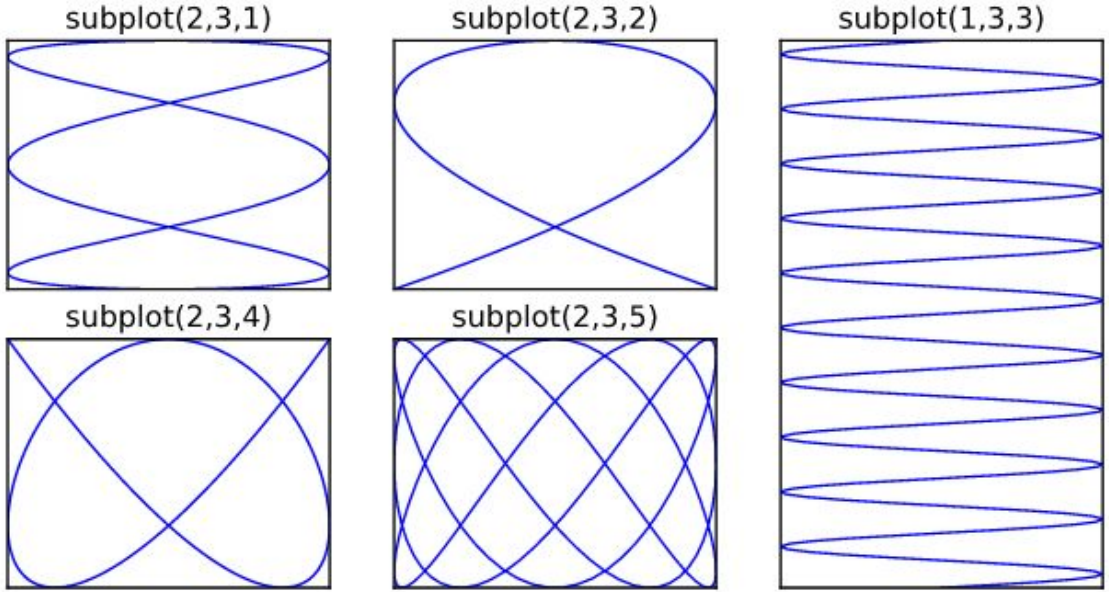
- Todos os gráficos devem ser interpretáveis corretamente em preto e branco. Abusem de recursos que possibilitem isso (markers, linhas tracejadas, etc)





Dicas

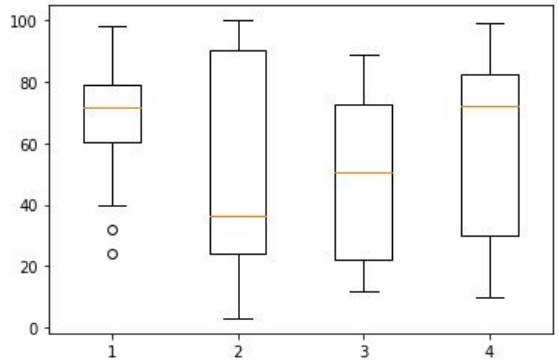
- Agrupar gráficos com o comando subplot() deixa o trabalho mais organizado

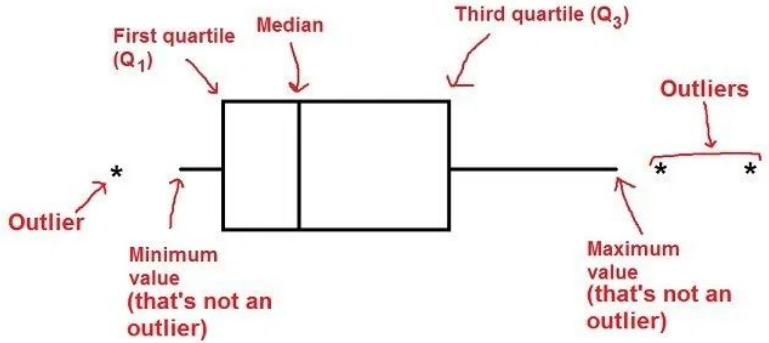


Dicas

- Pesquisem tipos diferentes de gráficos. O boxplot, por exemplo, é excelente para mostrar as estatísticas de uma amostra (mediana, max., min., etc.)

```python
import matplotlib.pyplot as plt

sample1 = [82,76,24,40,67,62,75,78,71,32,98,89,78,67,72,82,87,66,56,52]
sample2 = [62,5,91,25,36,32,96,95,3,90,95,32,27,55,100,15,71,11,37,21]
sample3 = [23,89,12,78,72,89,25,69,68,86,19,49,15,16,16,75,65,31,25,52]
sample4 = [59,73,70,16,81,61,88,98,10,87,29,72,16,23,72,88,78,99,75,30]

box_plot_data = [sample1, sample2, sample3, sample4]
plt.boxplot(box_plot_data)
plt.show()
```





https://drive.google.com/open?id=1XgIFOByBGyFQDULnOchc9IBBADGWrC69

Dicas

- Sempre que um gráfico for utilizado, ele deve ser referenciado e descrito textualmente

- Antes de representar seus resultados, faça os seguintes questionamentos:
  - Gráfico vai resolver seu problema? Para poucos dados, por que não tabela?
  - Tabela vai resolver seu problema? Talvez 1 frase seja suficiente (pouquíssimos dados)
  - Quanto espaço ainda há disponível para o trabalho?
  - Que tipo de variável eu preciso representar?
    - Qualitativa / Quantitativa / Discreta / Contínua?

Tipos de Variáveis

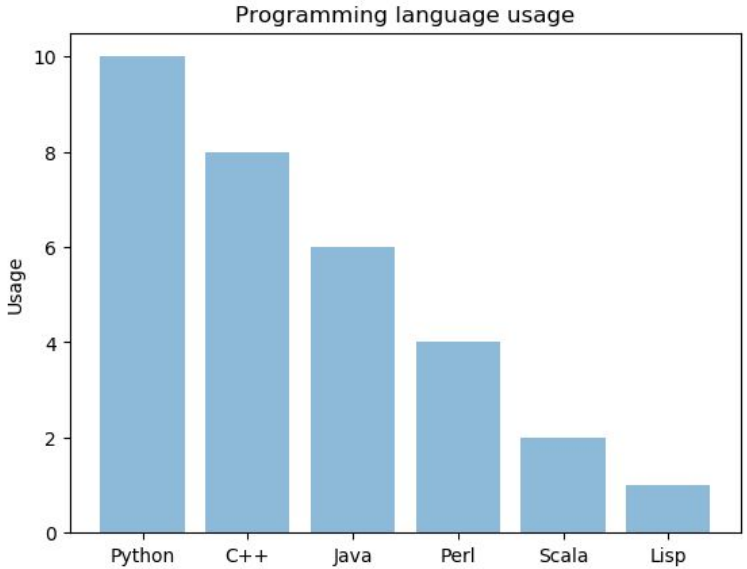
- Qualitativa: agrupa os dados em categorias
  - Ordenada
    - Ex:. faixas etárias (criança, pré-adolescente, adolescente, adulto, idoso)
  - Não ordenada
    - Ex: áreas do conhecimento: biologia, matemática, computação, física, etc

- Quantitativa: representa os dados na forma de valores numéricos
  - Discreta
    - Ex.: número de usuários simultâneos de internet numa universidade
  - Contínua
    - Ex.: tempo de execução de um processo

Tipos de Variáveis

- Variáveis qualitativas: melhor representadas por barras ou polar chart
  - No caso de variáveis ordenadas, gráficos de barra mostram o sentido de ordenação
  - Ex. (barras):

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
5  y_pos = np.arange(len(objects))
6  performance = [10,8,6,4,2,1]
7
8  plt.bar(y_pos, performance, align='center', alpha=0.5)
9  plt.xticks(y_pos, objects)
10 plt.ylabel('Usage')
11 plt.title('Programming language usage')
12
13 plt.show()
```

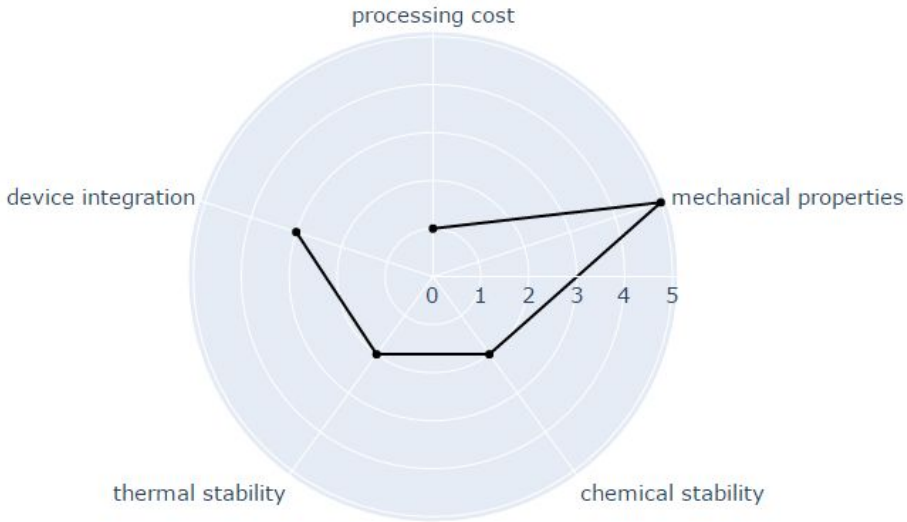


https://drive.google.com/open?id=1ur-fimxH1-yZ9-INpIOgCah_o8_YzHo4

Tipos de Variáveis

- Variáveis qualitativas: melhor representadas por barras ou polar chart
  - No caso de variáveis ordenadas, gráficos de barra mostram o sentido de ordenação
  - Ex. (polar):

```python
import plotly.graph_objects as go

fig = go.Figure(data=
    go.Scatterpolar(
        r=[1, 5, 2, 2, 3],
        theta=['processing cost','mechanical properties','chemical
               'thermal stability', 'device integration'],
        mode = 'lines+markers',
        line_color = 'black'
    ))

fig.update_layout(
    showlegend = False,
    font_size = 15,
    polar = dict(
        angularaxis = dict(
            direction = "clockwise")
    ),
)
fig.show()
```



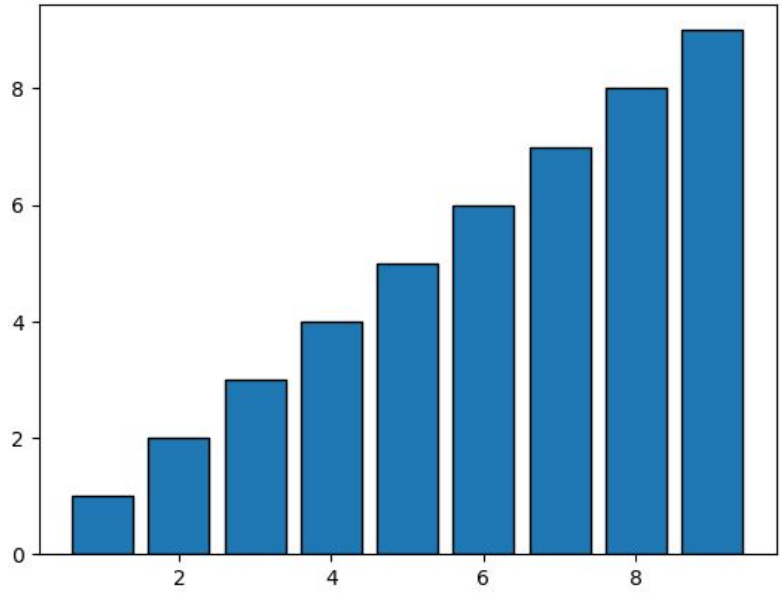
https://drive.google.com/open?id=1bf4nGHNozid33C1-vAunhBH13IgZ3MOb

Tipos de Variáveis

- Variáveis quantitativas: são bem representadas em gráficos do tipo X-Y
  - Variáveis discretas são adequadas a gráficos de pontos ou gráficos de barras
  - Ex. (barras):

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  fig, ax = plt.subplots(num='mwe_scalar_bar_ec')
5  x = np.arange(1, 10)
6  ax.bar(x, x, edgecolor="black")
7  plt.show()
```

https://drive.google.com/open?id=1hg9DvswvoPCsxxgcafHeGrYahx9-VYET

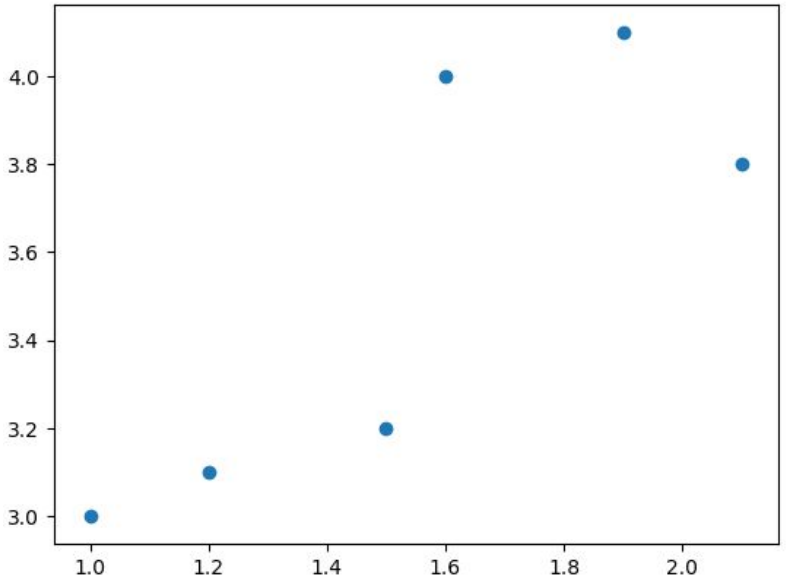


Tipos de Variáveis

- Variáveis quantitativas: são bem representadas em gráficos do tipo X-Y
  - Variáveis discretas são adequadas a gráficos de pontos ou gráficos de barras
  - Ex. (pontos):

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x_axis = [1, 1.2, 1.5, 1.6, 1.9, 2.1]
5  y_axis = [3, 3.1, 3.2, 4, 4.1, 3.8]
6
7  plt.scatter(x = x_axis, y = y_axis)
8  plt.show()
```

https://drive.google.com/open?id=1dQE4DpUVi31ZixAeYDn0XznIFQQ6oSTR

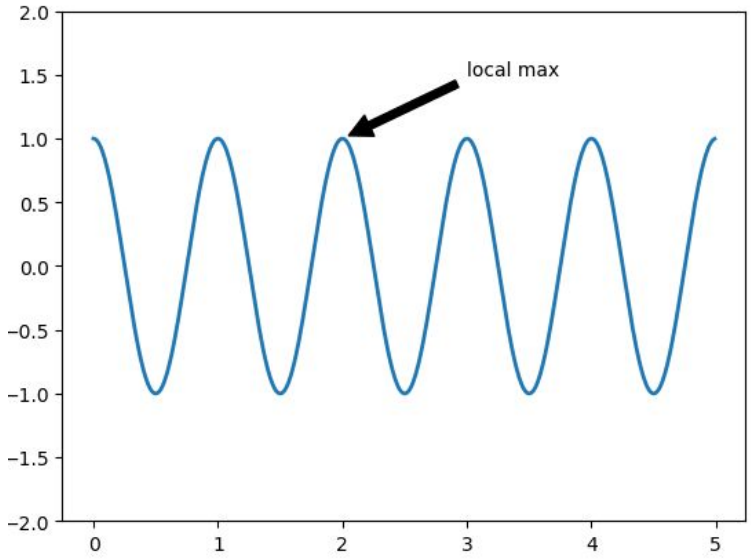


Tipos de Variáveis

- Variáveis quantitativas: são bem representadas em gráficos do tipo X-Y
  - Variáveis discretas são adequadas a gráficos de pontos ou gráficos de barras
  - Variáveis contínuas se adequam a gráficos de linhas
  - Ex. (linhas):

```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
             arrowprops=dict(facecolor='black', shrink=0.05),
             )

plt.ylim(-2, 2)
plt.show()
```

https://drive.google.com/open?id=1zhKiqT_-zvvYGjAzg9U-85VIBEDn2_ig

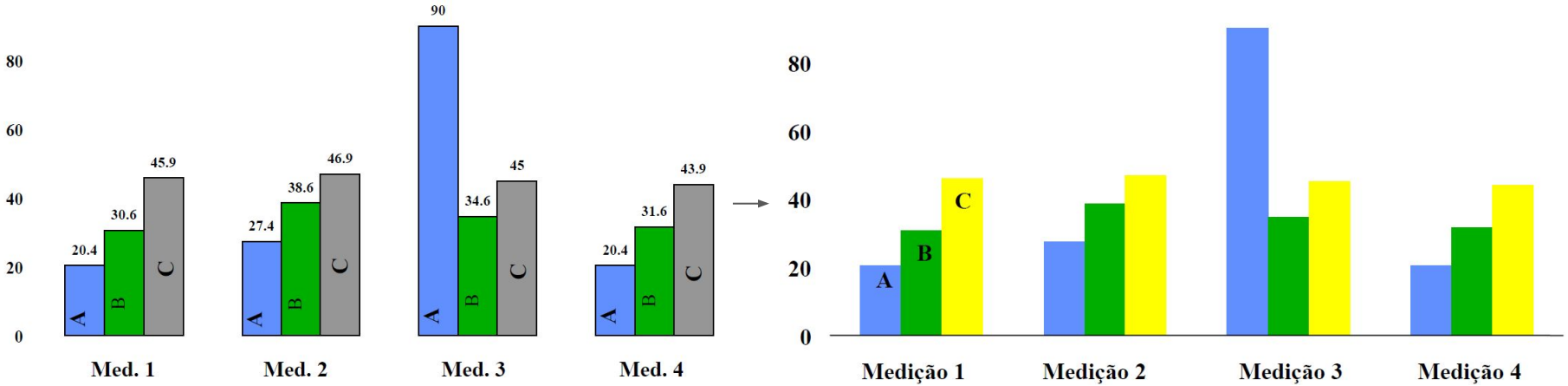


Princípios para Bons Gráficos

- Maximize a relação dados/tinta

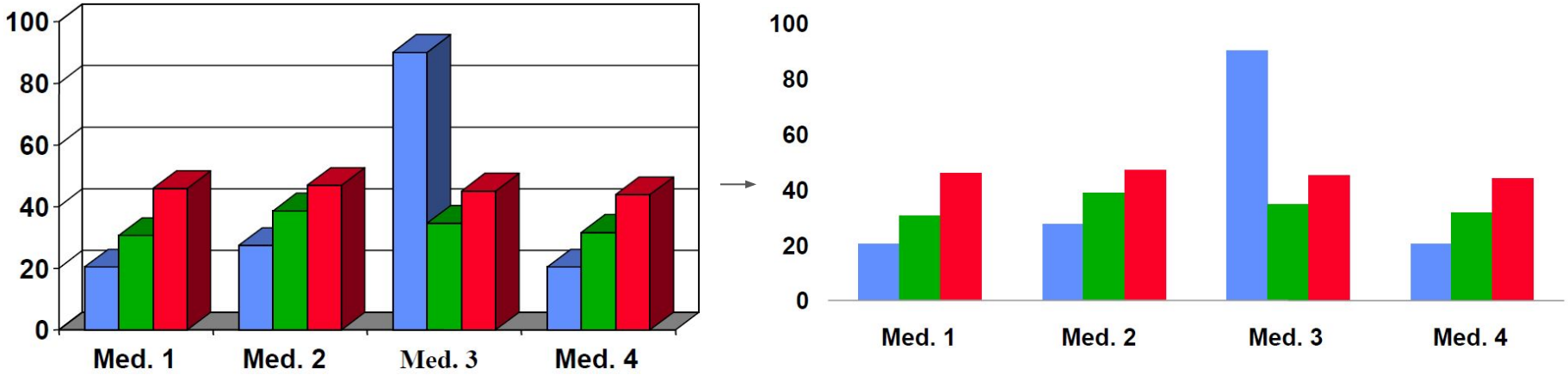
$$\text{Data-Ink Ratio} = \frac{\text{Data-Ink}}{\text{Total ink used in graphic}}$$

- Diminua tinta que não representa dados, elimine tinta de dados redundantes



Princípios para Bons Gráficos

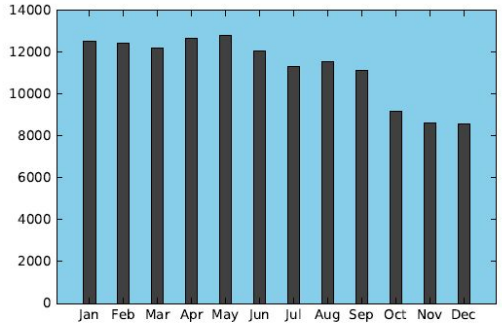
- Cuidado com o excesso de “estilo”. Ir direto ao ponto é ideal para os leitores dos seus resultados



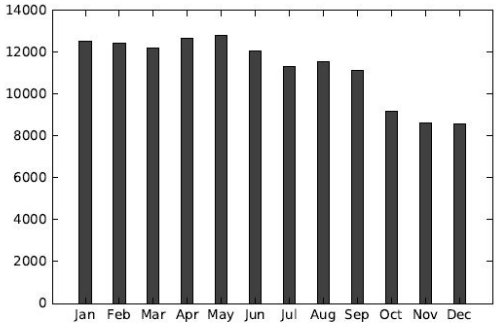
Princípios para Bons Gráficos

- Resumindo: evitar tudo que não são resultados ou elementos explicativos

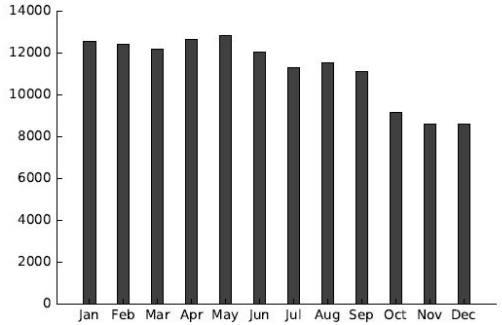
1



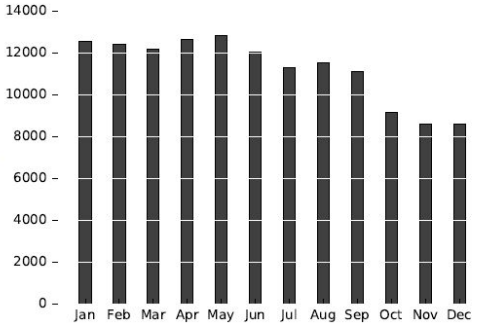
2



3



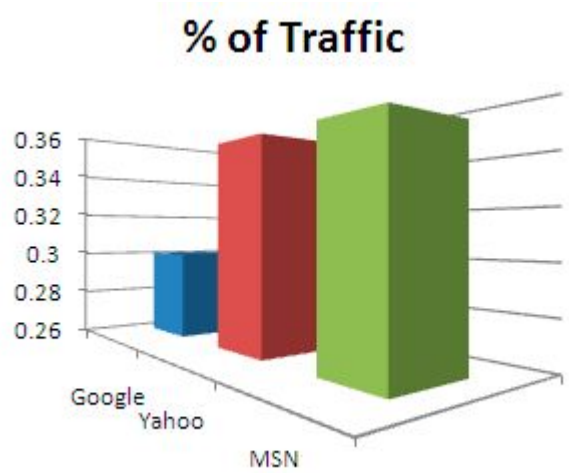
4

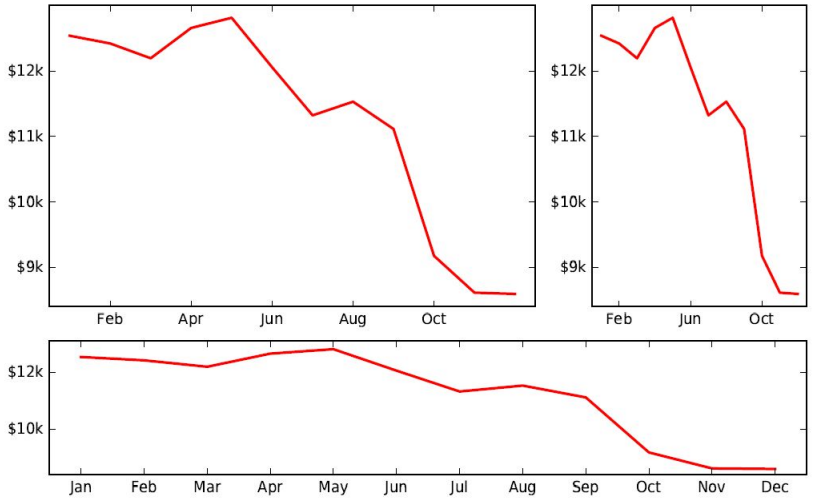


Princípios para Bons Gráficos

- Jamais use escalas e ângulos de imagem para favorecer ou esconder resultados indesejados (minimizar *lie factor*)

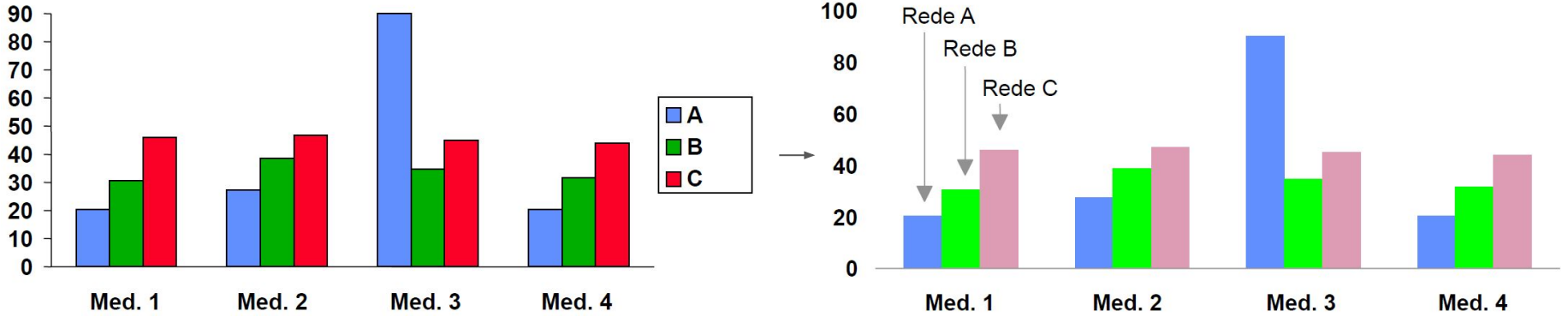
$$\text{lie factor} = \frac{(\text{size of an effect in the graphic})}{(\text{size of the effect in the data})}$$





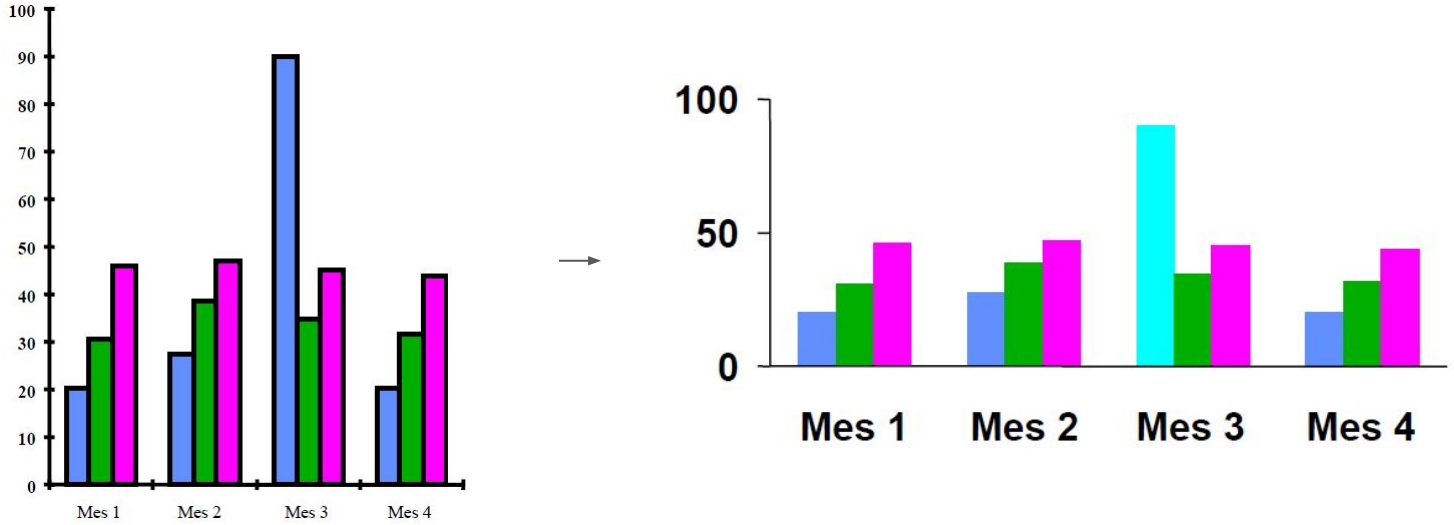
Princípios para Bons Gráficos

- Coloque rótulos de forma inteligente e autocontida
- Evite o uso das cores verde e vermelho no mesmo gráfico



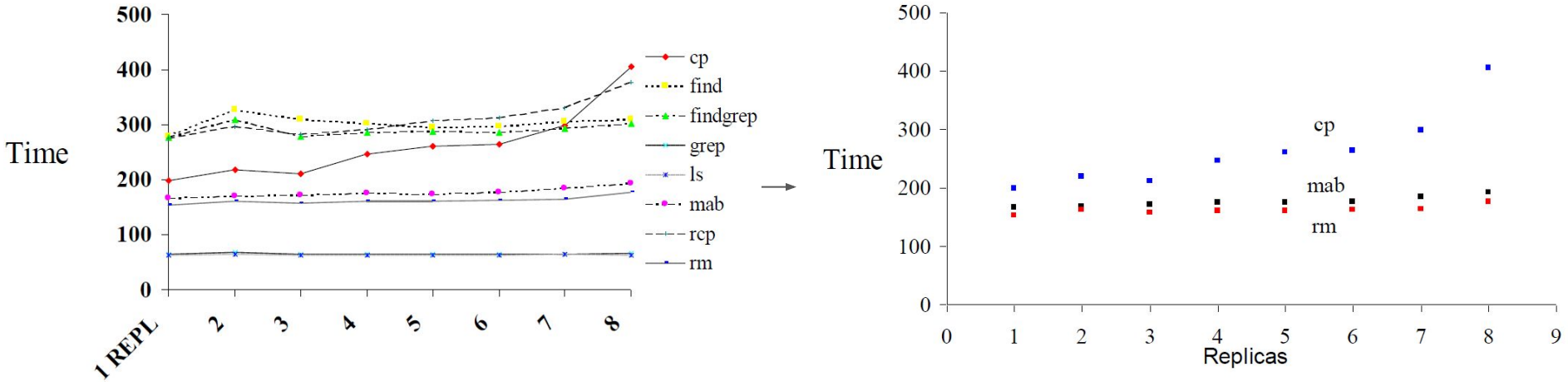
Princípios para Bons Gráficos

- Linhas finas e ausência de contornos ajudam a estética
  - Use linhas grossas e contornos para realçar alguma informação específica
- A dimensão horizontal deve ser maior que a vertical: até 50% maior



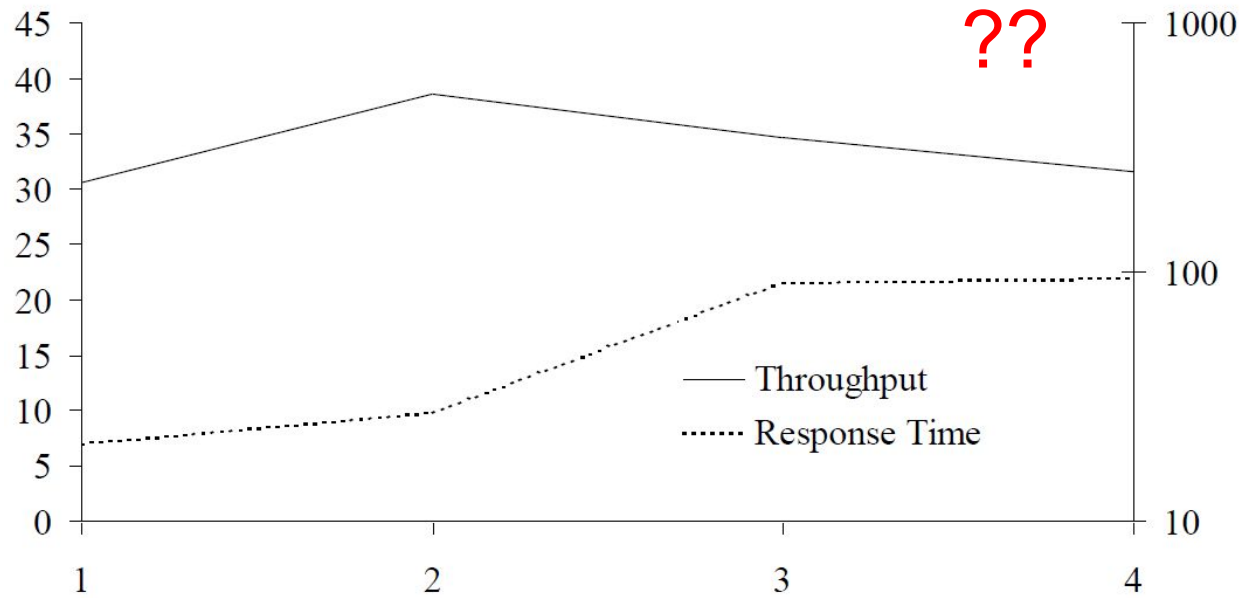
Princípios para Bons Gráficos

- Evitar excesso de informação num único gráfico
  - Máximo de 6 curvas num único gráfico de linhas
  - Máximo de 10 barras num único histograma (gráfico de barras)
  - Máximo de 8 fatias num gráfico em pizza (polar chart)



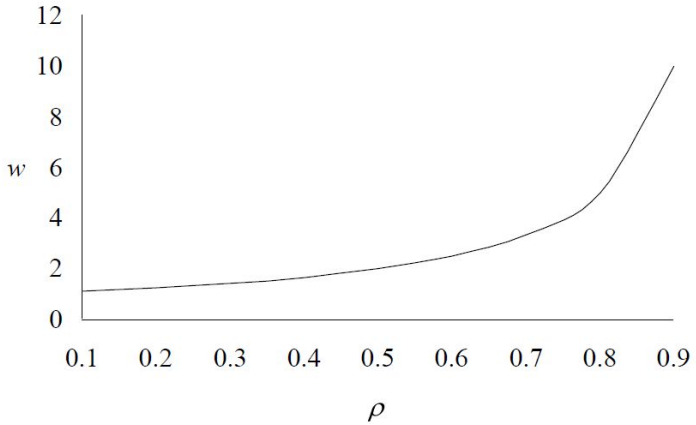
Princípios para Bons Gráficos

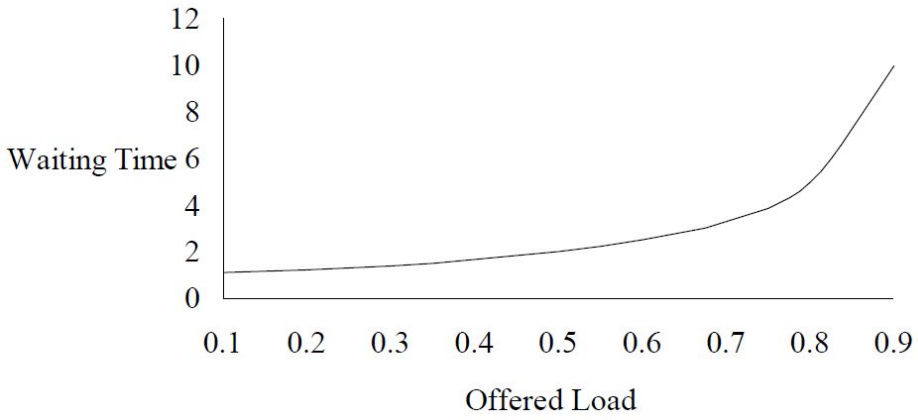
- Evitar o uso de múltiplas escalas num mesmo gráfico
  - Representar bananas com bananas e maçãs com maçãs



Princípios para Bons Gráficos

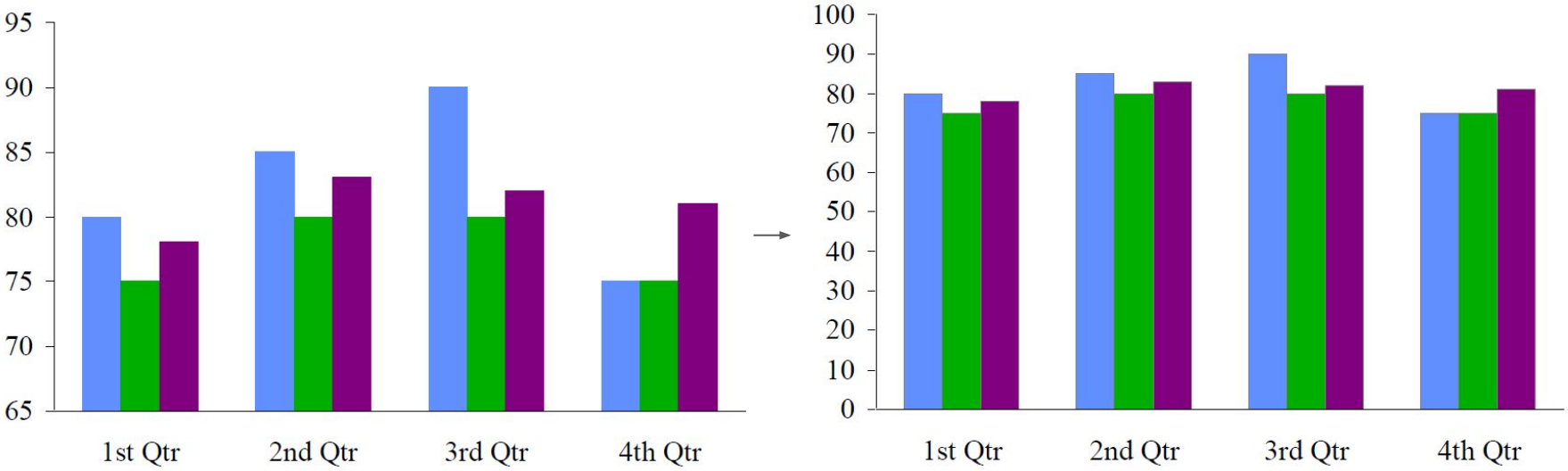
- Gráficos devem ser auto-explicativos
  - Dêem preferência para textos ao invés de símbolos





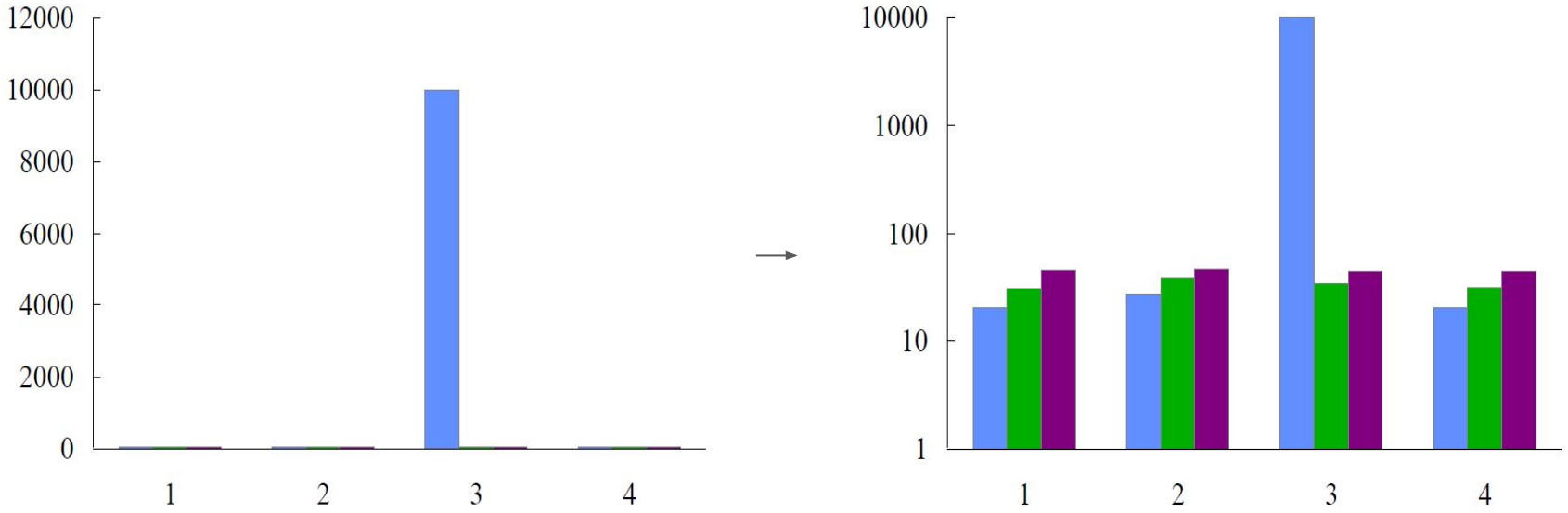
Princípios para Bons Gráficos

- Evitar plotar gráficos com origem diferindo de zero
  - Se o resultado não ficar bom (vide exemplo abaixo), talvez valha a pena mudar a escala (intervalos no eixo y ou log)



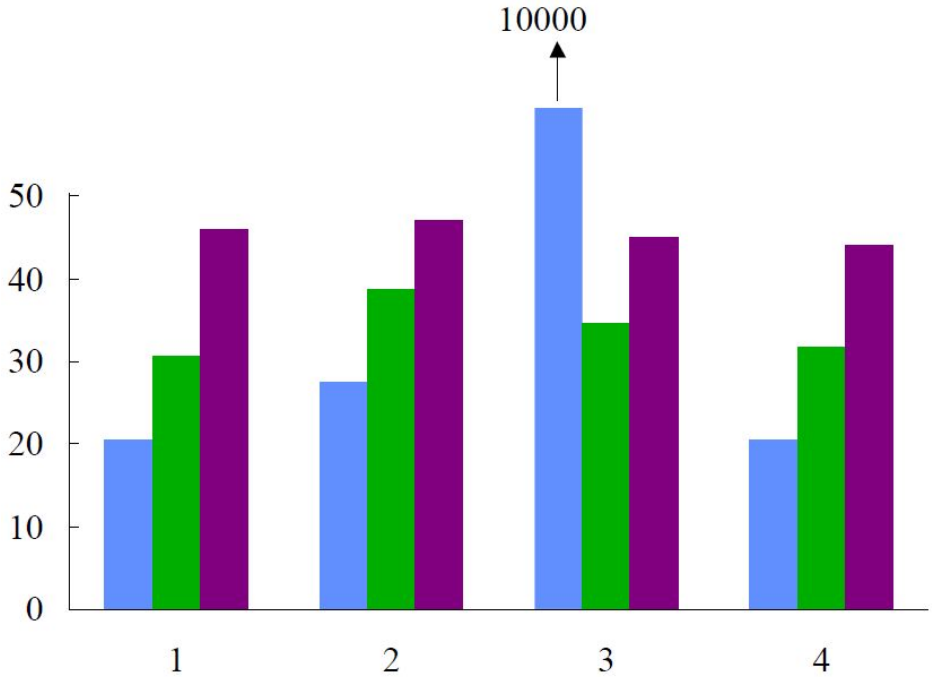
Princípios para Bons Gráficos

- Escala logarítmica pode ajudar a evidenciar resultados de difícil visualização
  - Quando as curvas ficarem muito coladas ou estranhas, tentem usar semilogx ou semilogy



Princípios para Bons Gráficos

- Quando algum dado é muito maior que os demais, vale a pena truncar



Princípios para Bons Gráficos

- Evite ligar pontos caso as interpolações não sejam válidas
  - Quando os intervalos no eixo X são grandes, pode ser perigoso simplesmente ligar pontos
  - Suavizar curvas (eliminar quinas) não é uma boa prática