



**Cruzeiro do Sul
Virtual**
Educação a Distância

CONCEITOS GERAIS

Prof. Ismar Frango



Visualização da Informação – Para quê?

A humanidade, mesmo antes de desenvolver a escrita, usou de representações pictóricas (imagens, figuras, etc.) para representar o mundo à sua volta. Os desenhos rupestres encontrados em cavernas ao redor do mundo nos mostram a busca por formas de representação da informação de maneira que fosse compreensível ao longo do tempo e por várias pessoas, preferencialmente vindas de distintas culturas.

Obviamente, com a evolução da humanidade e, especialmente nos últimos anos, com o avanço das tecnologias digitais, as técnicas de representações visuais para a visualização da informação cresceu enormemente.

No mundo empresarial, no jornalismo, na Educação, nas mais diversas áreas da pesquisa e do desenvolvimento humano, saber utilizar e selecionar os métodos adequados para visualizar informações que advêm de uma crescente massa de dados – especialmente no contexto de Ciência de Dados – requer o domínio de algumas técnicas, que iremos estudar aqui.

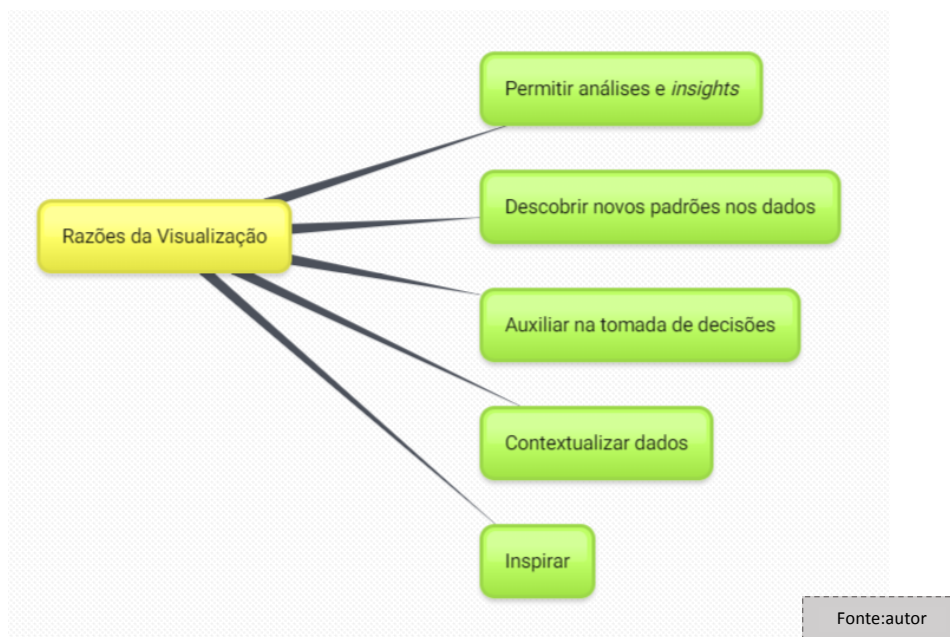
Veremos que toda visualização depende de um **propósito**, podendo ser direcionada para um **público específico**; e que as técnicas de visualização estão intimamente relacionadas à **natureza da informação**.

Fonte <https://www.flickr.com/photos/franciscoferreira/3981941089> - Licença CC-BY

Objetivos da Visualização

Para que visualizar? Esta pergunta em um mundo em que, no ano de 2020, é possível que haja um total de 40 trilhões de gigabytes de dados, com 2,2 milhões de terabytes de novos dados gerados todos os dias, parece inocente, mas é importante para frisar o papel da Visualização da Informação. Com essa imensa quantidade de dados, é virtualmente impossível para um ser humano tentar encontrar padrões, tomar decisões e ter *insights* sobre qualquer massa de dados sem a aplicação de técnicas corretas de visualização dos mesmos.

Resumidamente:



Um exemplo clássico desse fenômeno é conhecido por Quarteto de Ascombe: são quatro *datasets* que têm muitas similaridades entre si (mesmas médias, desvios padrões, etc), mas cujos dados possuem comportamentos muito distintos entre si, o que só é possível perceber quando visualizados.

Os *datasets* numerados de I a IV, com duas variáveis cada, x e y, são o Quarteto de **Ascombe**:

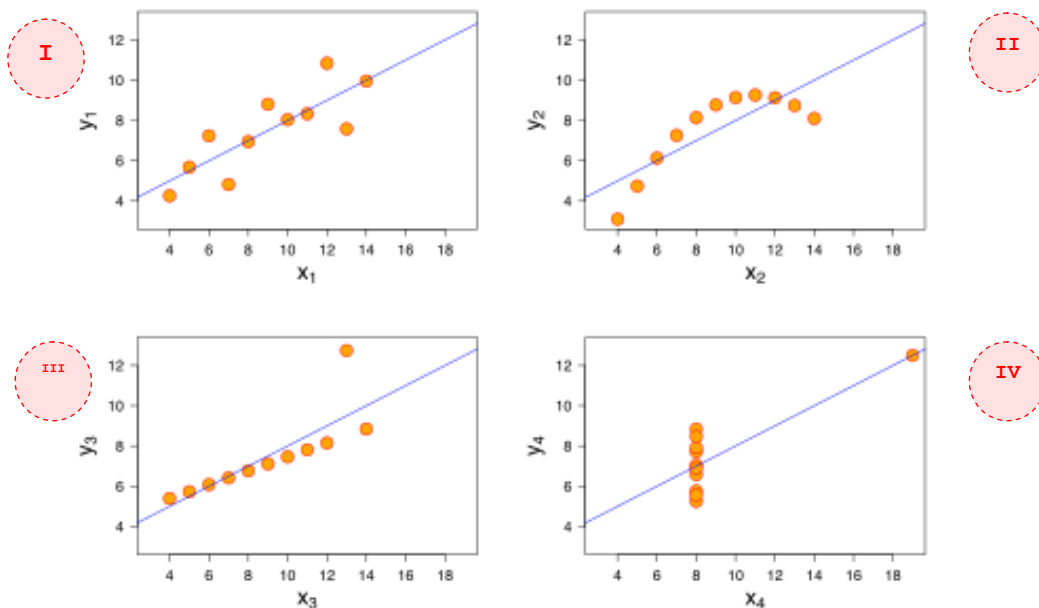
Nome em homenagem ao estatístico inglês Francis Ascombe, que propôs esse quarteto em 1973, para mostrar que apenas a Estatística Descritiva nem sempre é suficiente para compreender as relações entre os valores de um *dataset*.

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	9.13	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

Qualquer um dos *datasets* (I, II, III e IV) tem os mesmos valores

Propriedade	Valor
Média de x	9
Variância de x:	11
Média de y	7.50
Variância de y:	4.125
Correlação entre x e y	0.816
Reta de regressão linear	$y = 3.00 + 0.500x$
Coefficiente R^2 de regressão	0.67

Porém, quando os *datasets* são plotados em um gráfico de dispersão (*scatter*), é que se percebe que são *datasets* cujos dados têm comportamentos diferentes.

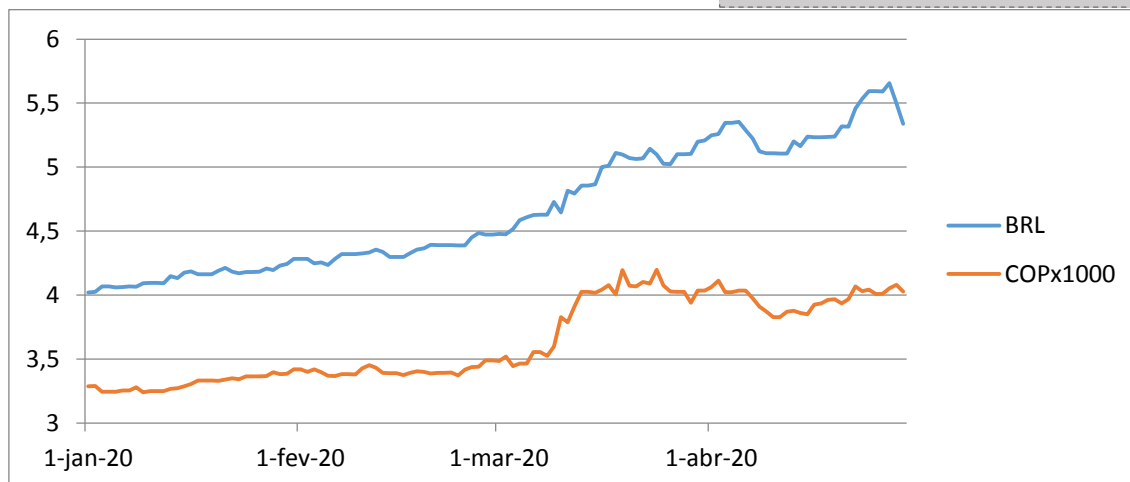


Fonte https://en.wikipedia.org/wiki/Anscombe%27s_quartet - Licença CC-BY-SA

Um outro exemplo: observe o seguinte trecho de *dataset* com os valores da cotação do dólar estadunidense em relação ao Real brasileiro (BRL) e ao Peso colombiano (COP), nos 120 primeiros dias de 2020:

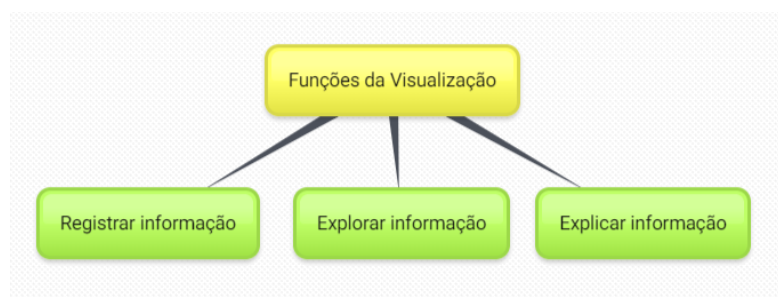
1-jan-20	2-jan-20	3-jan-20	4-jan-20	5-jan-20	6-jan-20	7-jan-20	8-jan-20	9-jan-20	10-jan-20	11-jan-20	12-jan-20	13-jan-20	14-jan-20	15-jan-20	16-jan-20	17-jan-20	18-jan-20	19-jan-20	20-jan-20	21-jan-20	22-jan-20	23-jan-20	24-jan-20	25-jan-20	26-jan-20	27-jan-20
4,02	4,03	4,07	4,07	4,06	4,06	4,07	4,06	4,09	4,10	4,10	4,09	4,15	4,13	4,18	4,18	4,16	4,16	4,16	4,19	4,21	4,18	4,17	4,18	4,18	4,18	4,18
3,29	3,29	3,25	3,25	3,24	3,26	3,26	3,28	3,24	3,25	3,25	3,25	3,27	3,27	3,29	3,31	3,33	3,33	3,33	3,33	3,34	3,35	3,34	3,37	3,37	3,36	3,36

A simples observação do dataset não nos permite a chegar a muitas conclusões a respeito dos dados. Há uma desvalorização de ambas as moeda? O quanto isto aparenta ser uma tendência? Ou é uma desvalorização momentânea? Os dados brutos não nos mostram com tanta clareza. Entretanto, uma visualização adequada facilita chegarmos a *insights* e tomarmos decisões:



O gráfico nos mostra que ambas as moedas seguem movimentos de altos e baixos mais ou menos similares até o início de março/2020, quando o COP sofre um salto abrupto de desvalorização, para em seguida se estabilizar em um platô ao redor de aproximadamente 4. Por outro lado, o BRL não teve uma desvalorização abrupta, mas seguiu desvalorizando continuamente até passar 5,5, para em seguida ter uma leve queda na desvalorização.

Resumidamente, a Visualização da Informação tem três funções básicas: registrar a informação (especialmente quando a informação já é de caráter visual); explorar a informação (no sentido de permitir a realização de análises sobre a informação) e explicar a informação (permitindo que a mesma seja compartilhada, que importantes aspectos dos dados sejam enfatizados e mesmo sendo usada como estratégia de persuasão).



Escolhendo gráficos e cores

A **escolha adequada do gráfico** mais aderente à natureza dos dados e ao propósito de visualização requer um bom conhecimento das estratégias possíveis de visualização, bem como entender a organização dos dados existentes em um *dataset*.

Já quanto às **cores**, é importante conhecer:

- Diferentes padrões, como RGB e HSL
- Conceito de paleta
- Conceitos de matriz (*hue*) e luminosidade

Iremos trabalhar esses conceitos quando os mesmos aparecerem em alguma visualização.



Conheça um pouco sobre os gráficos mais comuns em cada situação (em inglês):

<https://flourish.studio/2018/09/28/choosing-the-right-visualisation/>

<https://raw.githubusercontent.com/ft-interactive/chart-doctor/master/visual-vocabulary/poster.png>



Para ver erros comuns em escolhas de cores:

<https://serialmentor.com/dataviz/color-pitfalls.html>

Para saber sobre técnicas para escolhas de cores:

<https://medium.com/nightingale/how-to-choose-the-colors-for-your-data-visualizations-50b2557fa335>

Gráficos da Estatística Descritiva

A Estatística Descritiva básica se preocupa em sintetizar os dados da maneira mais direta possível. Para isso, ela lança mão de diversos gráficos que iremos estudar nesta unidade, como gráfico de barras e de setores (popularmente conhecido como “pizza”), por exemplo.

Iremos estudar cada uma dessas estratégias de visualização de informação mais básicas nesta unidade. A maioria delas está presente nas planilhas de cálculo e em aplicações online, de forma que não seria necessário programar em nenhuma linguagem para obter visualização da informação por meio desses gráficos básicos. Porém, já que para visualizações mais sofisticadas de *datasets* mais complexos – que é a realidade em Ciência de Dados, é

importante saber manipular bibliotecas de linguagens de programação que deem suporte a diferentes estratégias de visualização.

Várias linguagens de programação possuem bibliotecas com suporte bastante robusto a visualização da informação, como é o caso da linguagem R, da linguagem Javascript (especialmente por meio da biblioteca **D3.js**) e Python (por meio de módulos como `matplotlib`, `pandas` e `numpy`).



Saiba mais sobre D3.js em:
<https://d3js.org/>

Em nossos exemplos, iremos utilizar Python inicialmente. Para cada gráfico, apresentaremos uma breve introdução e um exemplo comentado de seu uso e implementação em Python.

Gráfico de barras



Sobre	O gráfico de barras é um dos tipos mais comuns para visualização da informação.
Uso	Use quando quiser mostrar a relação entre uma variável categórica e uma variável numérica . Sempre coloque um espaçamento entre as barras.
Mau uso	Quando é confundido com histograma
Curiosidade	Gráficos de barras verticais são geralmente chamados de gráficos de colunas , enquanto se chama gráfico de barras quando estão na horizontal. São, na realidade, a mesma representação visual – aqui, usaremos apenas o termo “barras”.

Fonte <https://python-graph-gallery.com/wp-content/uploads/BarBig-150x150.png> - Licença MIT

Exemplo

O *dataset* a seguir exibe o *ranking* do Spotify (app de músicas via *streaming*) para o Brasil no dia 31/12/2019. São mostradas as cinco primeiras posições.

Fonte <https://spotifycharts.com/regional/br/daily/2019-12->

	MÚSICA	ACESSOS
1	Liberdade Provisória by Henrique & Juliano	1.068.254
2	Sentadão by Pedro Sampaio	866.216
3	Combachy (feat. MC Rebecca) by Anitta	849.895
4	Surtada - Remix Brega Funk by Dadá Boladão	763.652
5	Cheirosa - Ao Vivo by Jorge & Mateus	758.198

O código a seguir mostra uma possível implementação em Python de uma visualização deste dataset por meio de um gráfico de barras. As linhas aparecem numeradas apenas para maior facilidade na explicação do código:

```

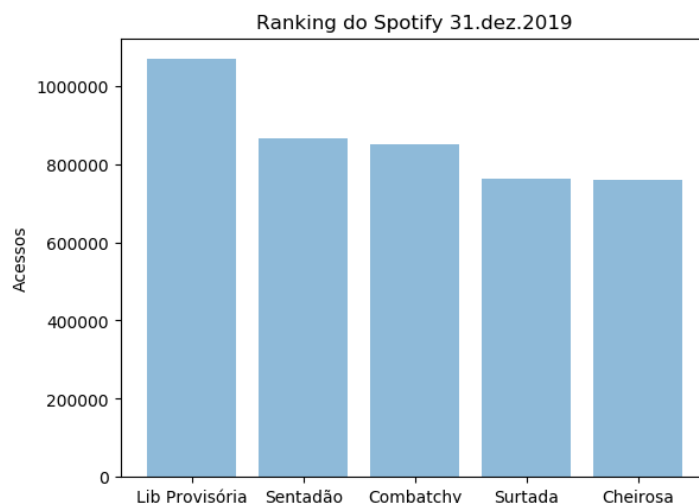
1. import matplotlib.pyplot as plt; plt.rcParams()
2. import numpy as np

3. musicas = ('Lib Provisória', 'Sentadão', 'Combachy', 'Surtada', 'Cheirosa')
4. indice = np.arange(len(musicas))
5. acessos = [1068254, 866216, 849895, 763652, 758198]

6. plt.bar(indice, acessos)
7. plt.xticks(indice, musicas)
8. plt.ylabel('Acessos')
9. plt.title('Ranking do Spotify 31.dez.2019')

10. plt.show()

```



Vamos analisar o código, linha a linha:


```
1. import matplotlib.pyplot as plt; plt.rcParams()
```

- `matplotlib` é a biblioteca do Python responsável por gráficos estáticos e dinâmicos. `pyplot` é o seu módulo mais simples. O uso de `import ... as` permite que usemos `plt` como um *alias* (apelido) para `matplotlib.pyplot`
- Cada vez que o `matplotlib` é carregado, ele define uma configuração de tempo de execução (*runtime configuration* - `rc`) contendo os estilos padrão para cada elemento de plotagem criado. A chamada `plt.rcParams()` mantém os estilos padrão (*defaults*)


```
2. import numpy as np
```

- A `numpy` é uma biblioteca do Python dedicada principalmente a realizar operações numéricas em *arrays* (coleções de valores). O uso de `import ... as` permite que usemos `np` como um *alias* (apelido) para `numpy`

```
3. musicas = ('Lib Provisória', 'Sentadão', 'Combatchy', 'Surtada', 'Cheirosa')
```

- Esta linha define um *array* chamado `musicas`. Seu conteúdo é o nome das cinco músicas do *dataset*

```
4. indice = np.arange(len(musicas))
```



Saiba mais sobre o método `arange` da biblioteca `numpy` em:
<https://realpython.com/how-to-use-numpy-arange/>

- O método `arange` de `numpy` (`np`) retorna um *array* numérico com uma sequência de números. Quando usado com apenas um parâmetro, este indica até antes de que número vai o *array*, começando em 0.
- É passado o valor de `len(musicas)` – o método `len` indica o tamanho de um *array* já existente (`musicas`) – no caso, 5. Neste caso, `indice` fica com o valor de `[0,1,2,3,4]`.

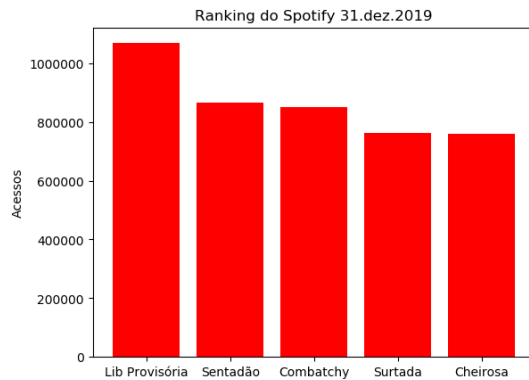
```
5. plt.bar(indice, acessos)
```


- O primeiro parâmetro é um *array* com a ordem das barras e o segundo com as alturas das barras (valores)

- Outros parâmetros podem ser usados neste método. Por exemplo, a inclusão do parâmetro `color` como em:

```
plt.bar(indice, acessos, color='red')
```

gera o seguinte gráfico:

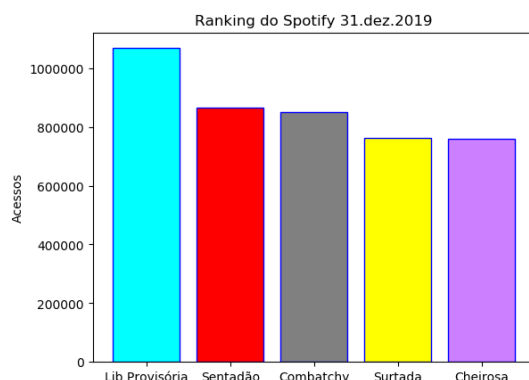



 Saiba mais sobre cores RGB normalizadas em:
http://doc.instantreality.org/tools/color_calculator/

- A cor do exemplo acima é dada por meio do seu nome (`color='red'`), mas podem ser usados valores **RGBA** normalizados (Red Green, Blue, Alpha, com valores de 0.0 a 1.0), ou mesmo um *array* com cores diferentes. Por exemplo, a chamada

```
plt.bar(y_pos, acessos, color=["cyan", "red", "gray", "yellow", (0.8, 0.5, 1.0, 1.0)], edgecolor='blue' )
```

resulta em um gráfico multicolorido, com o uso do argumento `edgecolor` para controlar a cor da borda:



 Conheça mais parâmetros do método bar em:
https://matplotlib.org/3.2.1/api/as_gen/matplotlib.pyplot.bar.html

```
7. plt.xticks(indice, musicas)
```

- O método `xticks` é responsável pela colocação de rótulos no eixo x (horizontal, eixo das abscissas). Recebe como parâmetros o *array*

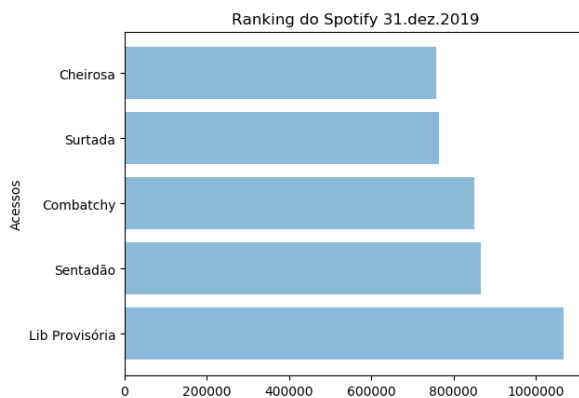
`indice` contendo a ordem dos valores e em seguida o `array` `musicas`, que contém as `strings` com os rótulos.

```
8. plt.ylabel('Acessos')
9. plt.title('Ranking do Spotify 31.dez.2019')
10. plt.show()
```

- A linha 8 usa o método `ylabel` para colocar um rótulo no eixo y (eixo das ordenadas, vertical)
- A linha 9 usa o método `title` para colocar um título no gráfico
- A linha 10 exibe o gráfico por meio do método `show`.

Uma pequena alteração no código (linhas 6 e 7) permite exibir as barras na horizontal, usando o método `barh` no lugar do `bar` e colocando o rótulo no eixo vertical, usando o método `yticks` no lugar do `xticks`:

```
6. plt.barh(indice, acessos, alpha=0.5)
7. plt.yticks(indice, musicas)
```



Para inverter a ordem das barras, basta mudar o `range` da linha 4, fazendo um `range decrescente`:

```
4. y_pos = np.arange(start=len(musicas), stop=0, step=-1)
```



Veja outros exemplos e mais opções de visualização de gráficos de barras em:

<http://python-graph-gallery.com/barplot>

Nível de instrução							
Sem instrução e menos de 1 ano de estudo	Ensino fundamental incompleto ou equivalente	Ensino fundamental completo ou equivalente	Ensino médio incompleto ou equivalente	Ensino médio completo ou equivalente	Ensino superior incompleto ou equivalente	Ensino superior completo ou equivalente	Não determinado
15.394	70.634	15.258	13.267	48.376	8.973	25.286	-
Fonte: IBGE - Pesquisa Nacional por Amostra de Domicílios Contínua trimestral							

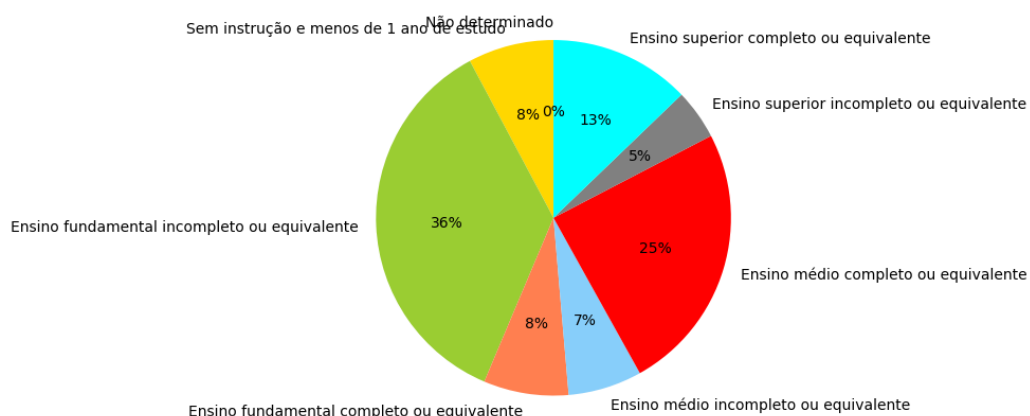
Uma implementação direta em Python seria:

```
1. import matplotlib.pyplot as plt
2. labels = 'Sem instrução e menos de 1 ano de estudo', 'Ensino fundamental
   incompleto ou equivalente', 'Ensino fundamental completo ou equivalente', 'Ensino
   médio incompleto ou equivalente', 'Ensino médio completo ou equivalente', 'Ensino
   superior incompleto ou equivalente', 'Ensino superior completo ou
   equivalente', 'Não determinado'
3. sizes = [15394, 70634, 15258, 13267, 48376, 8973, 25286, 0]
4. colors = ['gold', 'yellowgreen', 'coral', 'lightskyblue', 'red', 'grey', 'cyan',
   'blue']
5. plt.pie(sizes, labels=labels, colors=colors, autopct='%1.0f%%', startangle=90)
6. plt.axis('equal')
7. plt.show()
```

Veja que basta três elementos básicos para a implementação desse gráfico em Python:

- uma lista de rótulos, `labels`
- um *array* com os valores, `sizes`
- um *array* com as cores, `colors` (opcional)

Utilizou-se o método `pie`, que recebe esses três elementos como parâmetros e, no exemplo, indica que irá automaticamente identificar a porcentagem sem casas decimais (formato `%1.0f%%`) e que o ângulo de início será 90° - com as disposições dos elementos de maneira sequencial, no sentido anti-horário. O resultado (não muito bom) pode ser visto a seguir:



Claramente, há diversos problemas com este gráfico. Talvez o pior deles seja a disposição dos rótulos, que se sobrepõem em alguns casos. Uma legenda poderia ser pensada, como no exemplo:


```
import matplotlib.pyplot as plt

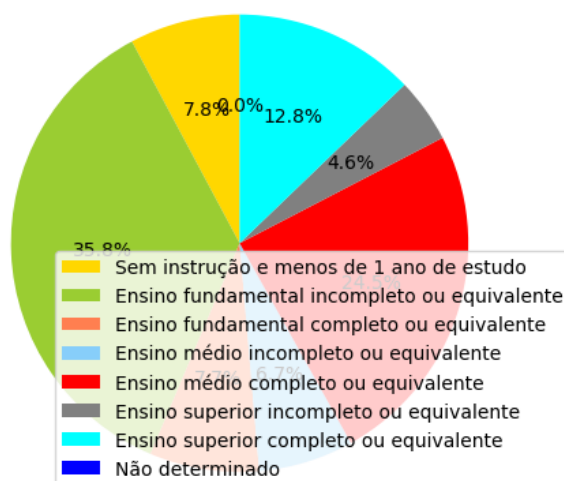
labels = 'Sem instrução e menos de 1 ano de estudo','Ensino fundamental incompleto ou equivalente','Ensino fundamental completo ou equivalente','Ensino médio incompleto ou equivalente','Ensino médio completo ou equivalente','Ensino superior incompleto ou equivalente','Ensino superior completo ou equivalente','Não determinado'
sizes = [15394, 70634, 15258, 13267, 48376, 8973, 25286, 0]
colors = ['gold', 'yellowgreen', 'coral', 'lightskyblue', 'red', 'grey', 'cyan', 'blue']

patches, texts, autotexts = plt.pie(sizes, colors=colors, autopct='%1.1f%%',
startangle=90)
plt.legend(patches, labels, loc="lower right")
plt.axis('equal')
plt.show()
```

O que este exemplo tem de diferente do anterior?

- A chamada ao método `pie` traz modificações sobre três elementos: `patches`, `texts`, `autotexts` respectivamente: a lista de setores dos gráficos, a lista de textos (que serão preenchidos com os labels) e os valores de cada setor.
- A chamada ao método `legend`, que cria a legenda

O resultado pode ser visto a seguir:



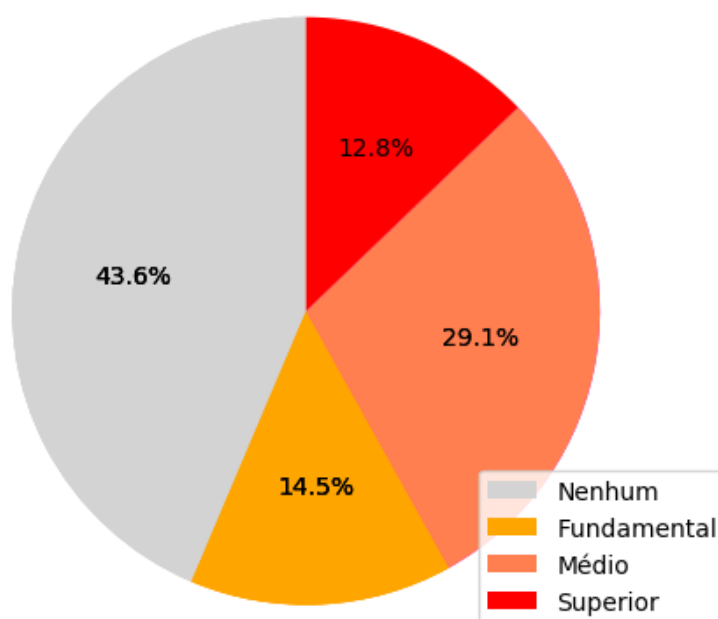
Claramente, ainda não é um bom resultado. O grande problema está na organização dos dados: há 8 setores no gráfico, sendo que um dos princípios da boa visualização de informação para este gráfico é trabalhar com o menor número de setores possível.

O que é o mais adequado, em termos de visualização? Claramente, reagrupar os dados de maneira a ter menos categorias – ou,

definitivamente, abandonar o gráfico de setores por outro tipo de visualização.

Note que nem sempre é possível o reagrupamento ou reestruturação dos dados. Se foram muitos valores categóricos, o gráfico de setores não será uma boa opção, certamente!

Como ficaria o gráfico se o *dataset* fosse categorizado por maior grau de instrução completo?



A visualização com menos categorias mostra-se mais adequada (embora, sem os rótulos de dados, ficaria difícil distinguir entre as categorias Fundamental e Superior). O código modificado segue:

```
import matplotlib.pyplot as plt

labels = 'Nenhum', 'Fundamental', 'Médio', 'Superior'
sizes = [86026, 28525, 57394, 25286]
colors = [ 'lightgray', 'orange', 'coral', 'red']

patches, texts, autotexts = plt.pie(sizes, colors=colors, autopct='%1.1f%%',
startangle=90)
plt.legend(patches, labels, loc="lower right")
plt.axis('equal')
plt.show()
```



Para saber mais sobre esse tipo de representação visual (*piechart*) em Python, visite:
https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.pie.html



Para saber mais, leia os capítulos iniciais dos e-books:

PERKOVIC, Ljubomir; VIEIRA, Daniel. Introdução à computação usando Python: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.

McKinney, W. Python Para Análise de Dados: Tratamento de Dados com Pandas, NumPy e IPython. São Paulo: Novatec, 2018