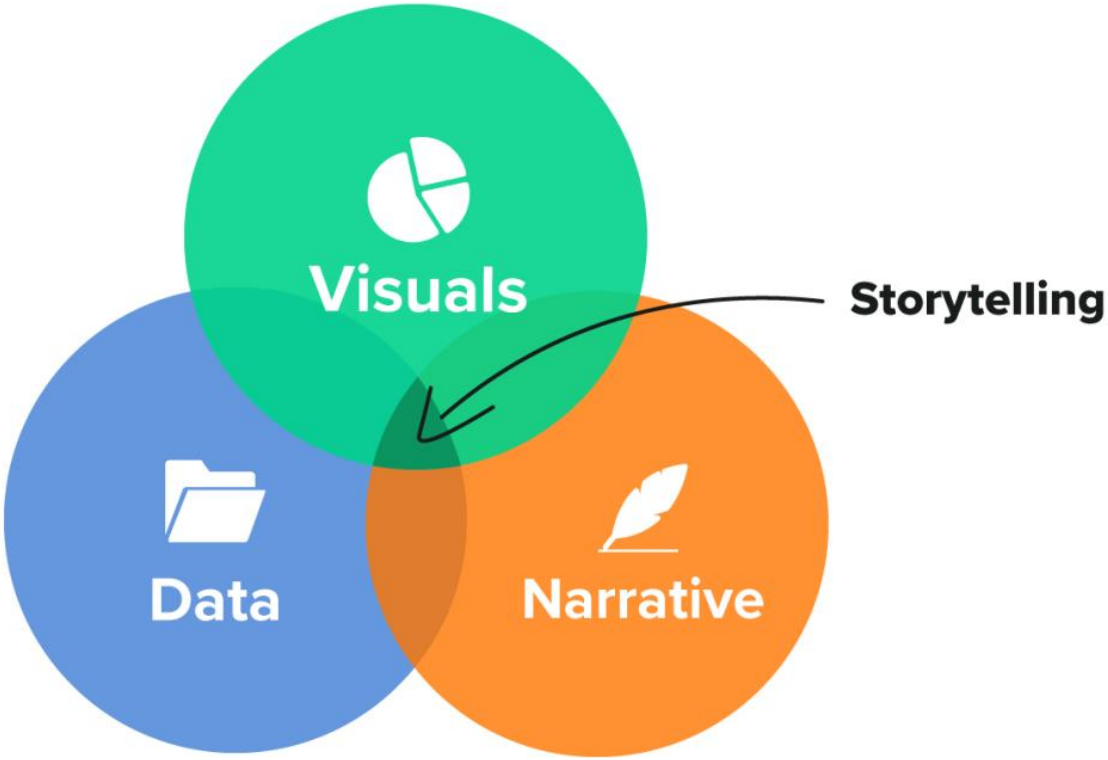
Aula 07

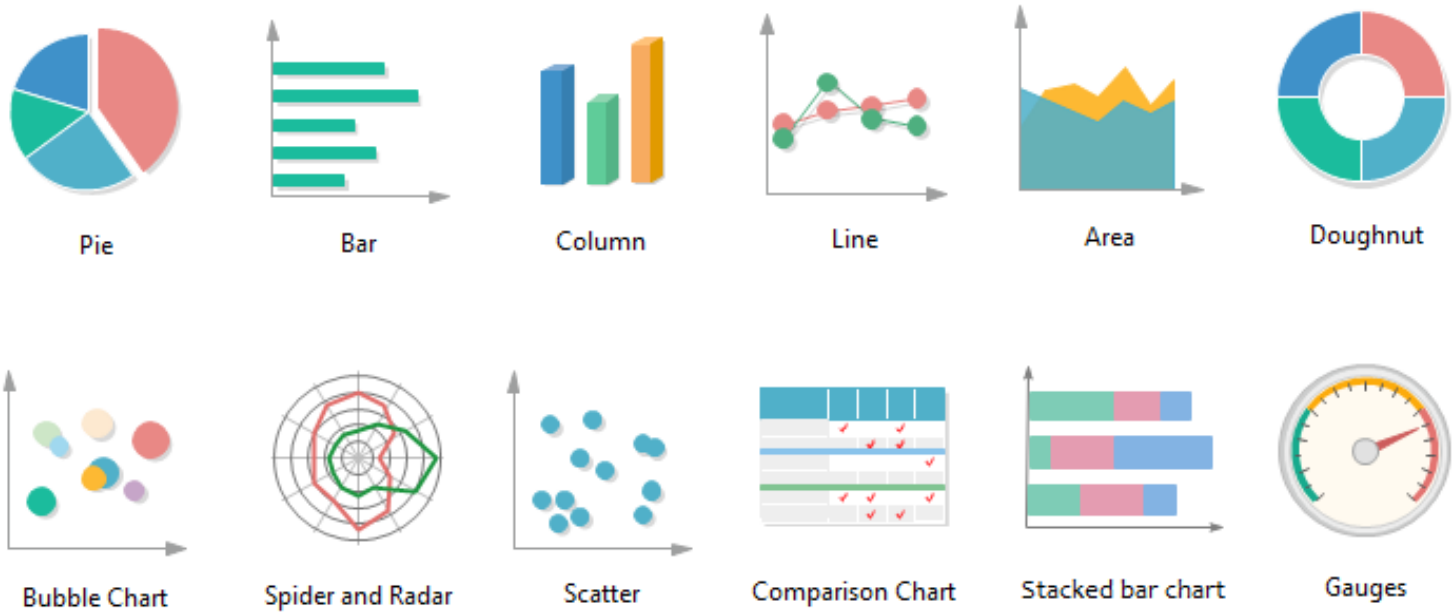
Como contar histórias?

Como contar histórias?

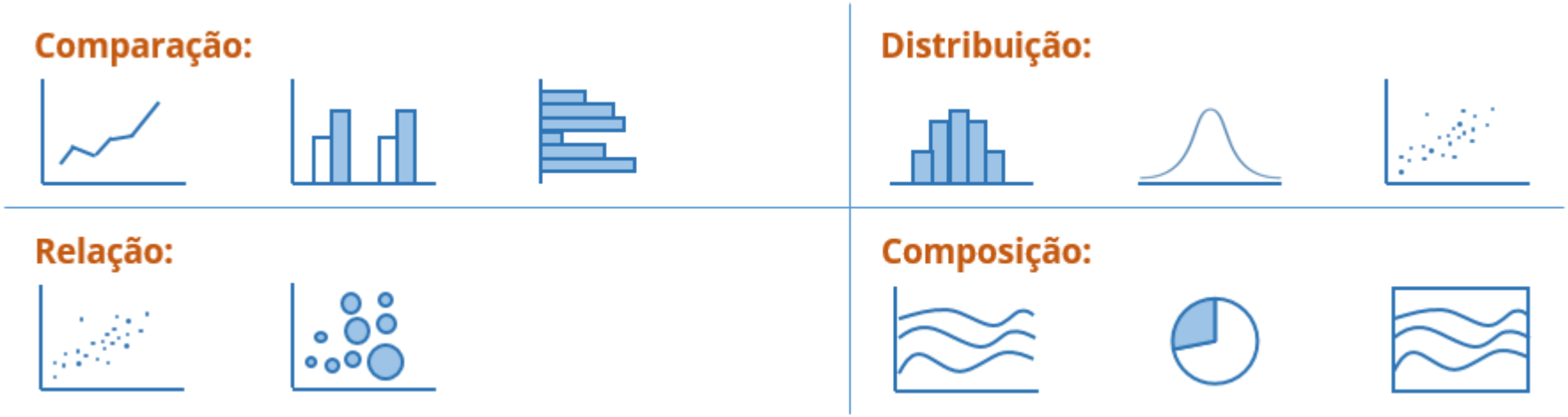
Como contar histórias com dados (data storytelling)?



[The Art of Data Storytelling To Engage and Persuade - AgencyAnalytics]


Pie
Bar
Column
Line
Area
Doughnut
Bubble Chart
Spider and Radar
Scatter
Comparison Chart
Stacked bar chart
Gauges

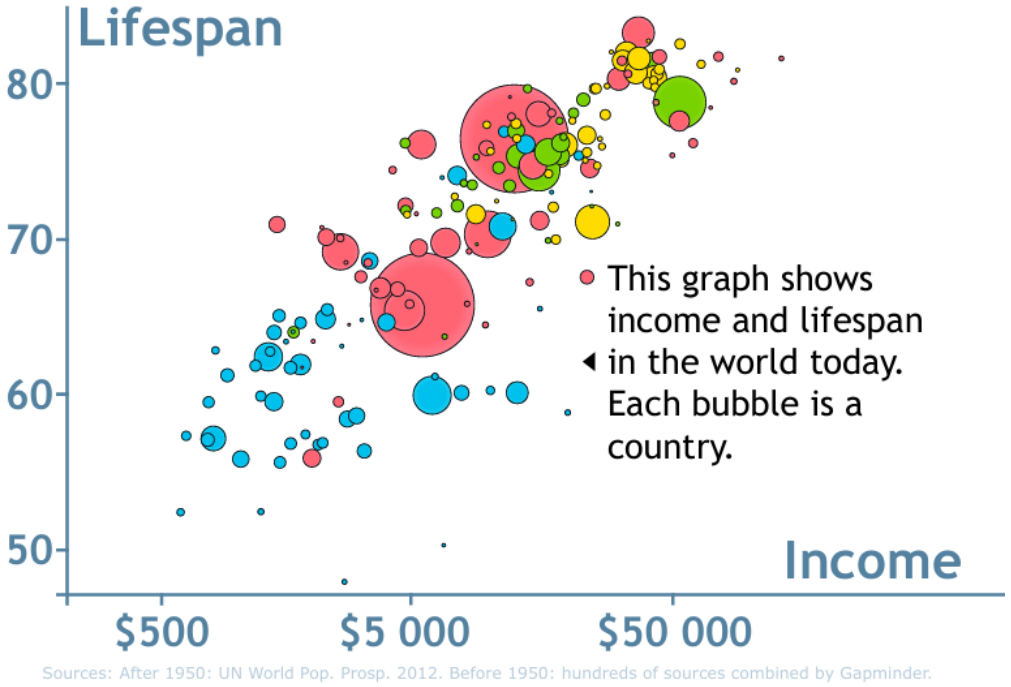
LinkedIn



[Como escolher o melhor gráfico para meus dados? — DataViz Basics: 2 de 4 | by DP6 Team | Blog DP6]

Exemplos:

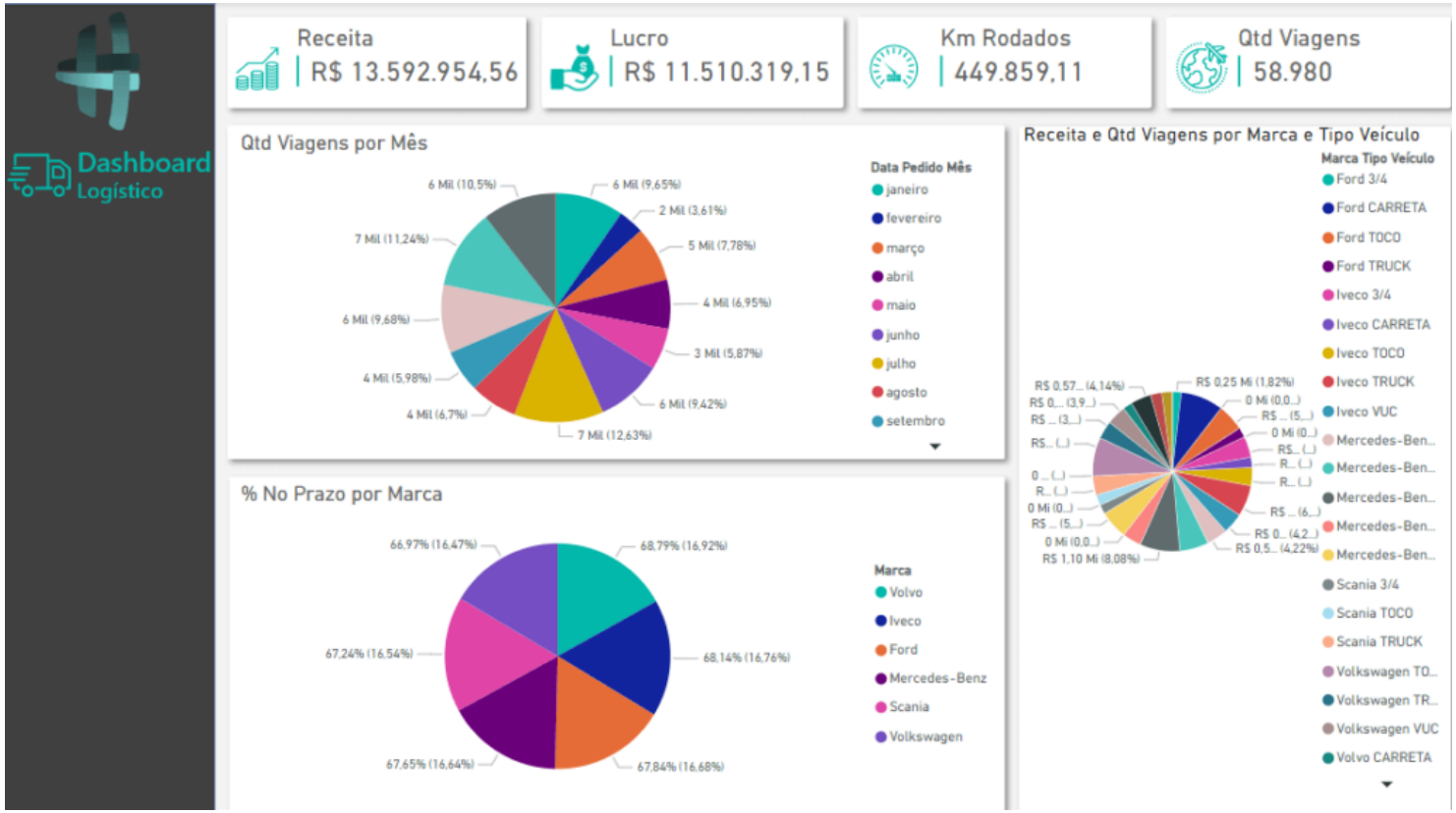
Como a renda se relaciona com a expectativa de vida?



[How Does Income Relate to Life Expectancy? | Gapminder]

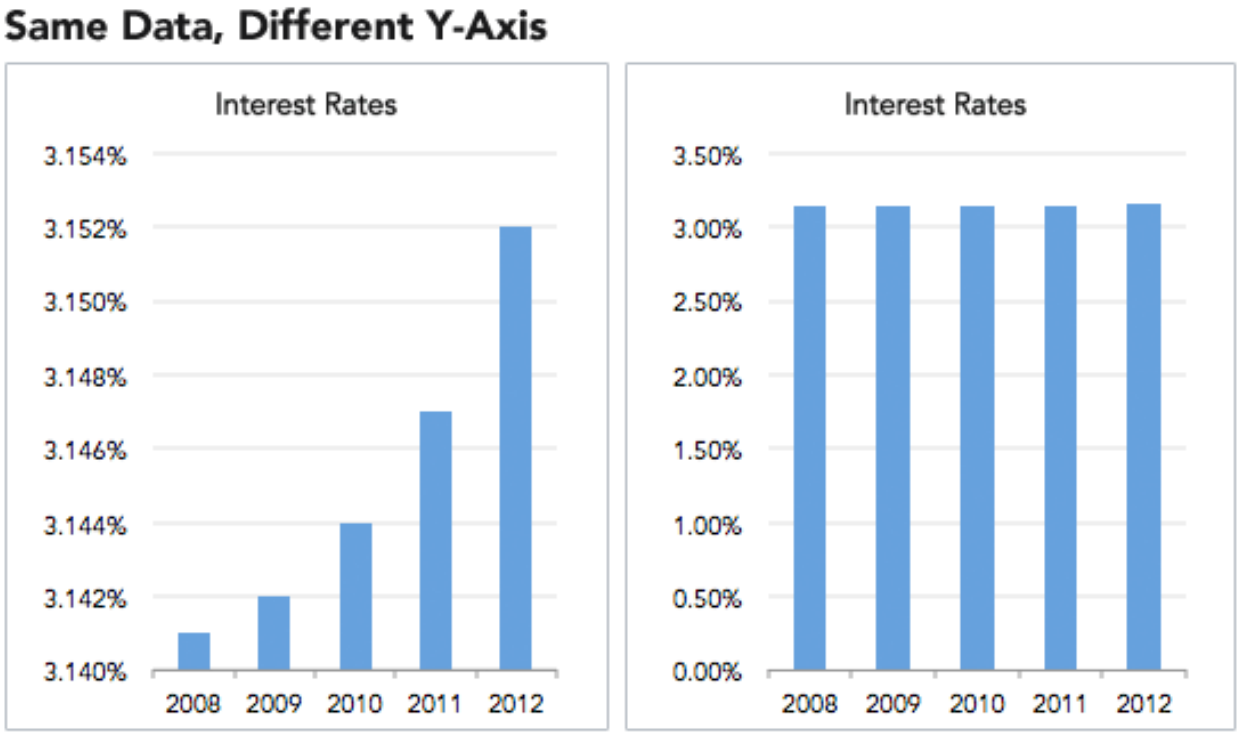
Exemplos:
Dashboard Logístico

Exemplos ruins:



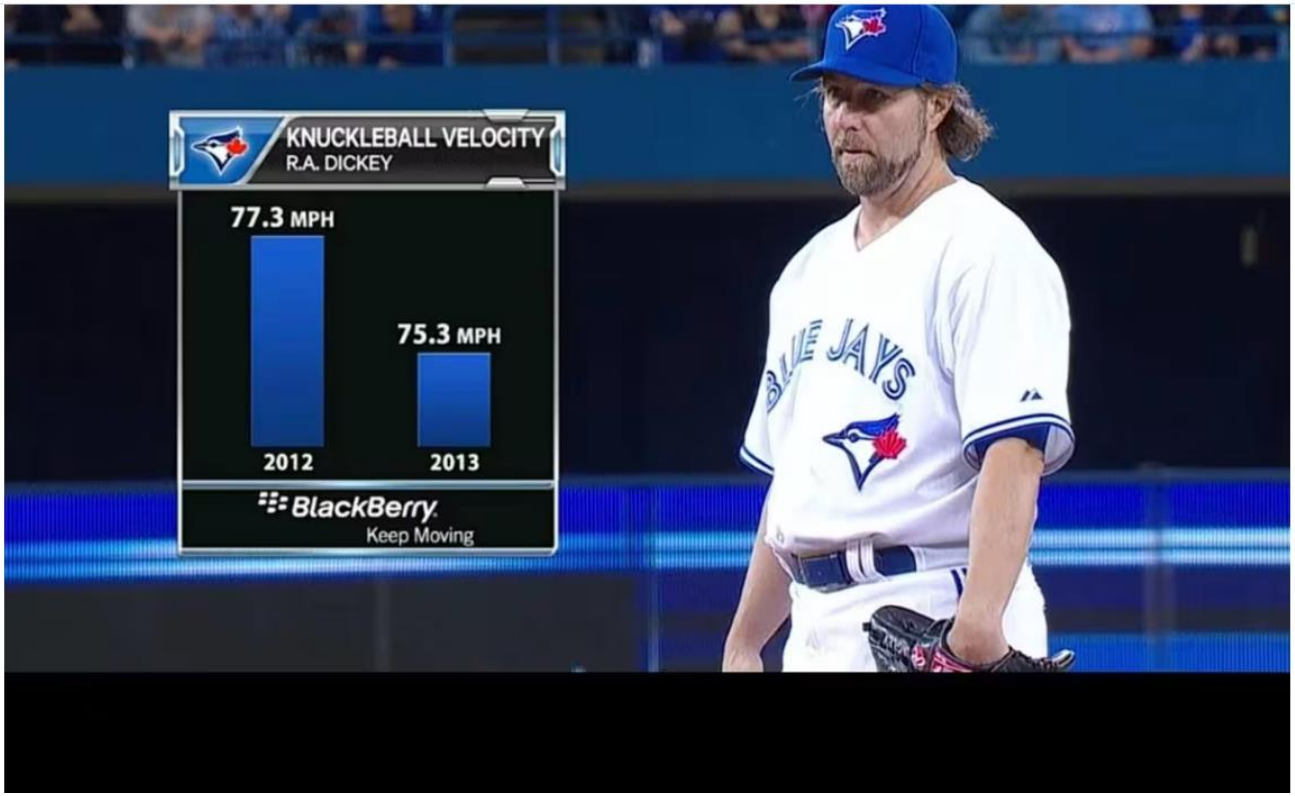
[O que não fazer no Dashboard – Dashboards Horríveis]

Como mentir com data viz



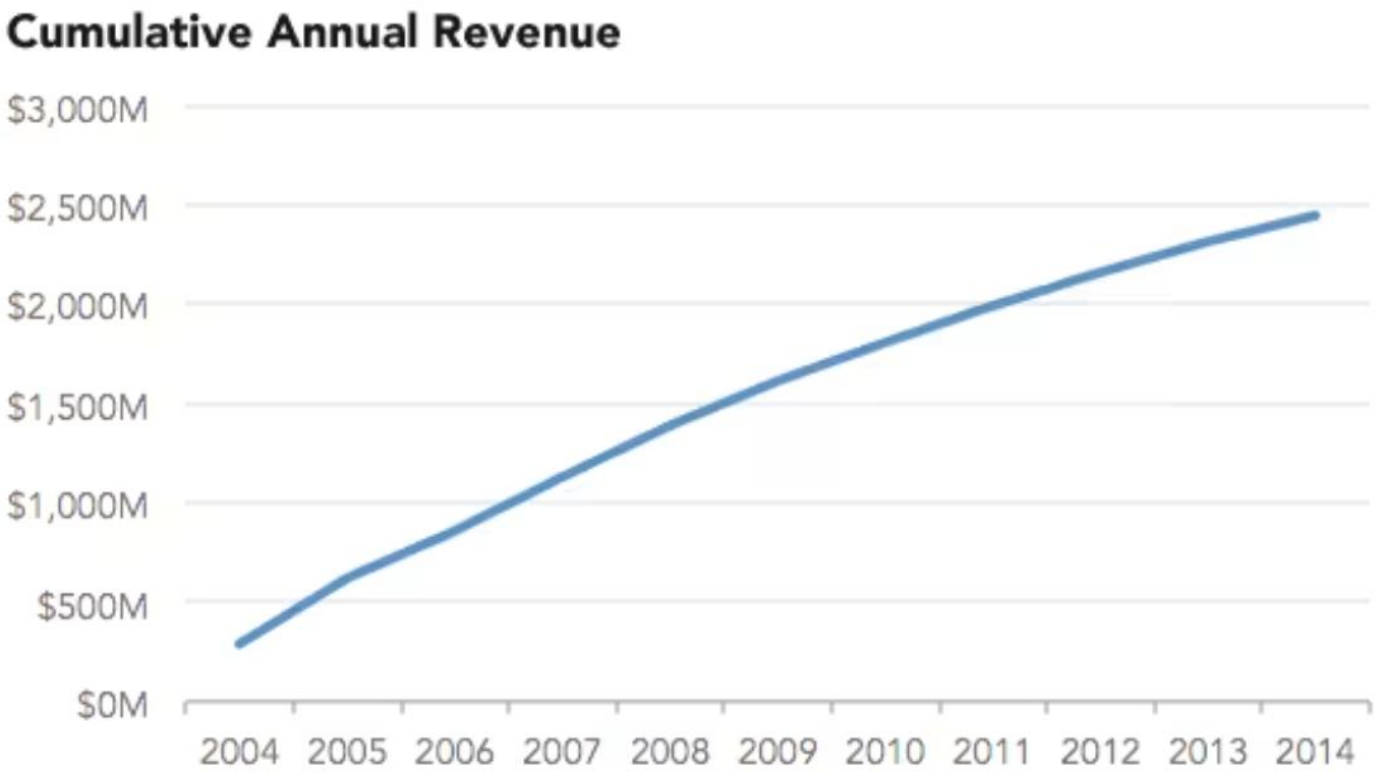
[How to Lie with Data Visualization | Heap]

Como mentir com data viz



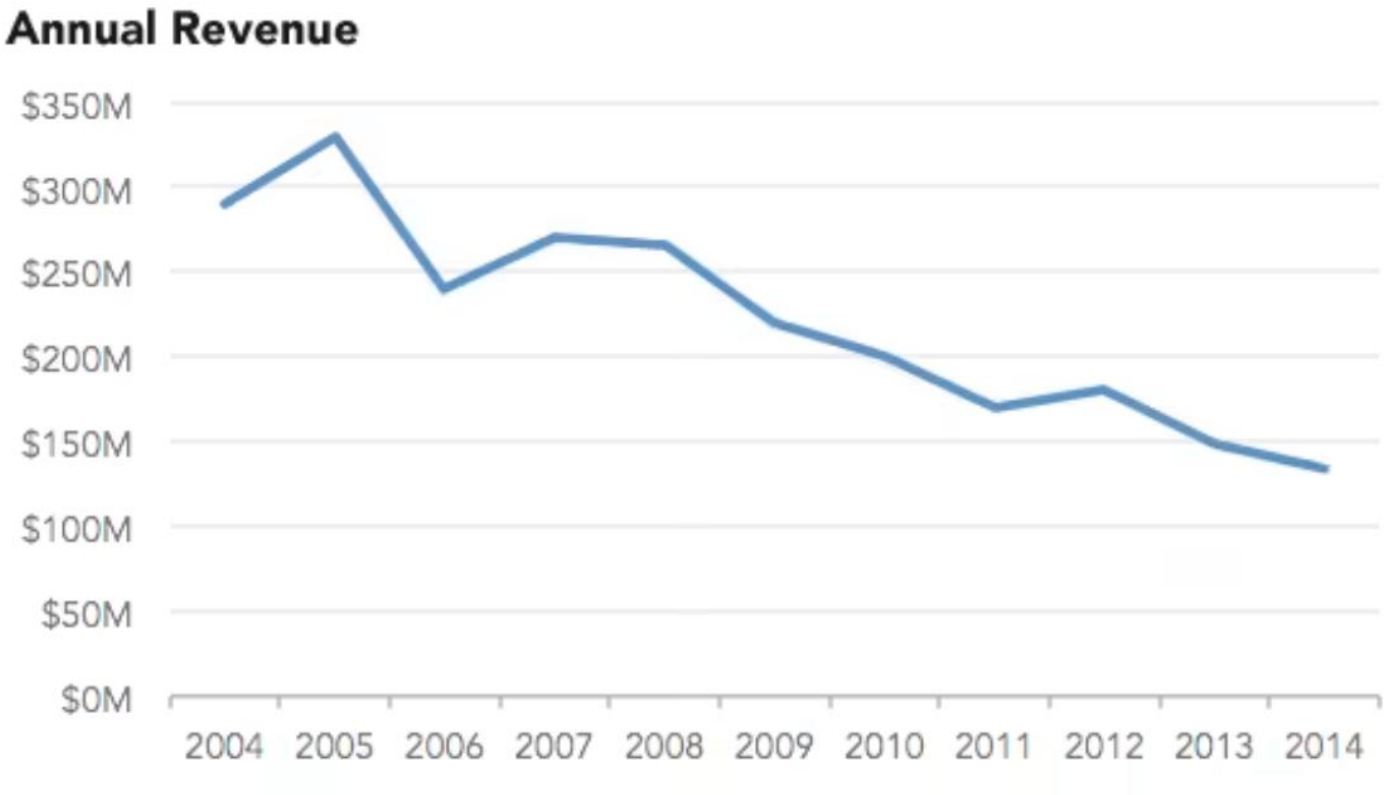
[How to Lie with Data Visualization | Heap]

Como mentir com data viz

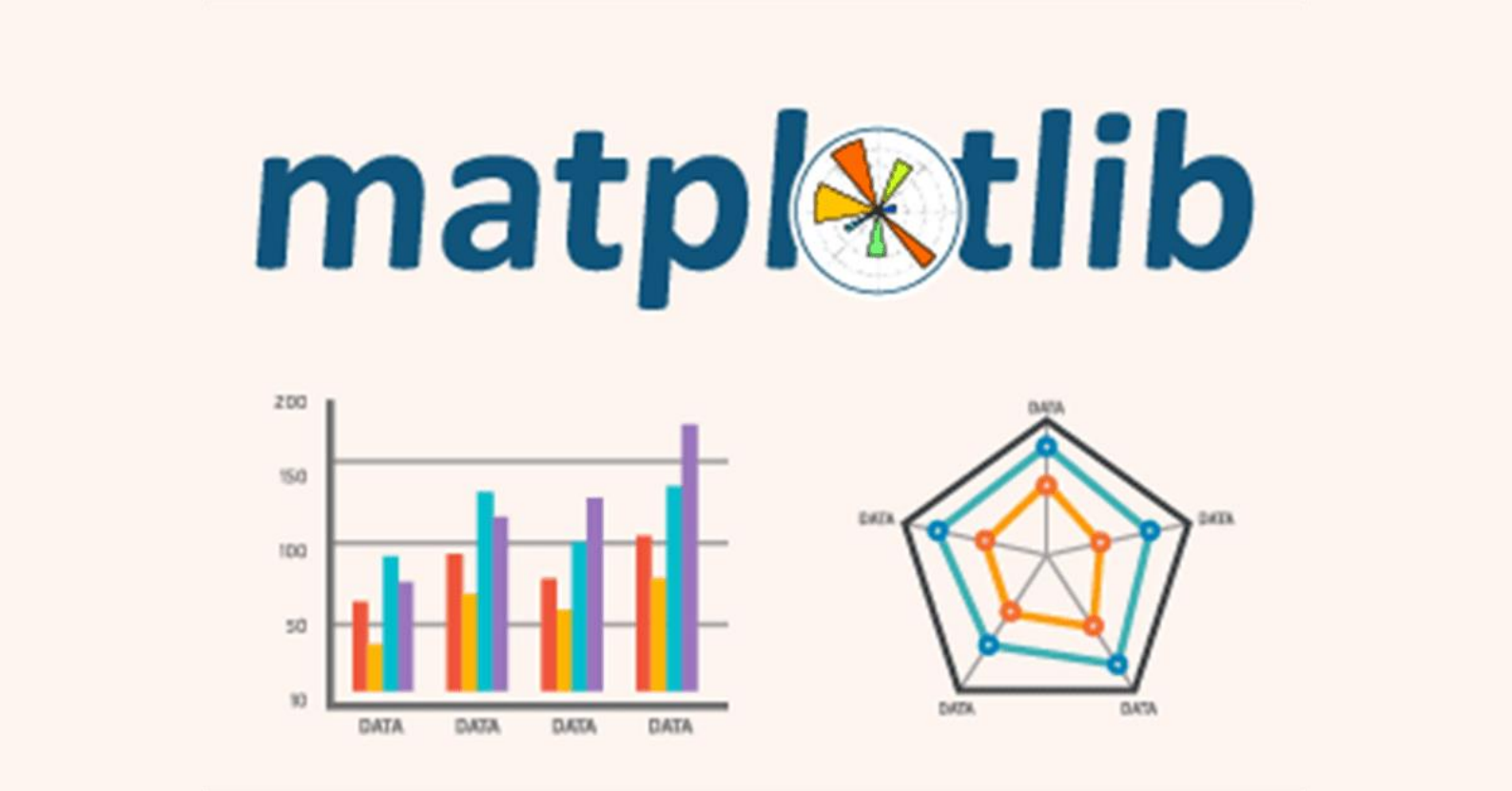


[How to Lie with Data Visualization | Heap]

Como mentir com data viz



[How to Lie with Data Visualization | Heap]



[Introduction to matplotlib : Types of Plots, Key features - 360DigiTMG]

matplotlib

Cheat sheet — Version 3.7.4

Quick start

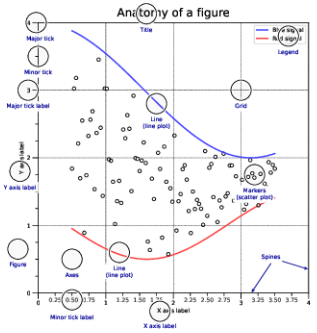
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
plt.show()
```

Anatomy of a figure



Subplots layout

```
subplot[s](rows, cols, …)
fig, axs = plt.subplots(3, 3)

G = gridspec(rows,cols, …)
ax = G[0, :]

ax.inset_axes(extent)

d=make_axes_locatable(ax)
ax = d.new_horizontal('10%')
```

Getting help

- matplotlib.org
- github.com/matplotlib/matplotlib/issues
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- https://gitter.im/matplotlib/matplotlib
- twitter.com/matplotlib
- Matplotlib users mailing list

Basic plots

- `plot([X], Y, [fmt], …)` — X, Y, fmt, color, marker, linestyle
- `scatter(X, Y, …)` — X, Y, [s]izes, [c]olors, marker, cmap
- `bar[h](x, height, …)` — x, height, width, bottom, align, color
- `imshow(Z, …)` — Z, cmap, interpolation, extent, origin
- `contour[f]([X], [Y], Z, …)` — X, Y, Z, levels, colors, extent, origin
- `pcolormesh([X], [Y], Z, …)` — X, Y, Z, vmin, vmax, cmap
- `quiver([X], [Y], U, V, …)` — X, Y, U, V, C, units, angles
- `pie(X, …)` — Z, explode, labels, colors, radius
- `text(x, y, text, …)` — x, y, text, va, ha, size, weight, transform
- `fill[_between][x](…)` — X, Y1, Y2, color, where

Advanced plots

- `step(X, Y, [fmt], …)` — X, Y, fmt, color, marker, where
- `boxplot(X, …)` — X, notch, sym, bootstrap, widths
- `errorbar(X,Y,xerr,yerr, …)` — X, Y, xerr, yerr, fmt
- `hist(X, bins, …)` — X, bins, range, density, weights
- `violinplot(D, …)` — D, positions, widths, vert
- `barbs([X], [Y], U, V, …)` — X, Y, U, V, C, length, pivot, sizes
- `eventplot(positions, …)` — positions, orientation, lineoffsets
- `hexbin(X, Y, C, …)` — X, Y, C, gridsize, bins

Scales

```
ax.set_[xy]scale(scale, …)
```

linear — any values
log — values > 0
symlog — any values
logit — 0 < values < 1

Projections

```
subplot(…, projection=p)
```

p='polar'
p='3d'
p=ccrs.Orthographic() — import cartopy.crs as ccrs

Lines

```
linestyle or ls
capstyle or dash_capstyle
```

"butt" "round" "projecting"

Markers

```
markevery
```

10 [0, -1] (25, 5) [0, 25, -1]

Colors

'Cn' 'x' 'name' (R,G,B[,A]) '#RRGGBB[AA]' 'x.y'

Colormaps

```
plt.get_cmap(name)
```

Uniform: viridis, magma, plasma
Sequential: Greys, YlOrBr, Wistia
Diverging: Spectral, coolwarm, RdGy
Qualitative: tab10, tab20
Cyclic: twilight

Tick locators

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_locator(locator)
ticker.NullLocator()
ticker.MultipleLocator(0.5)
ticker.FixedLocator([0, 1, 5])
ticker.LinearLocator(numticks=3)
ticker.IndexLocator(base=0.5, offset=0.25)
ticker.AutoLocator()
ticker.MaxNLocator(n=4)
ticker.LogLocator(base=10, numticks=15)
```

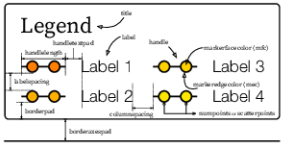
Tick formatters

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_formatter(formatter)
ticker.NullFormatter()
ticker.FixedFormatter(['zero', 'one', 'two', …])
ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)
ticker.FormatStrFormatter('>%d<')
ticker.ScalarFormatter()
ticker.StrMethodFormatter('{x}')
ticker.PercentFormatter(xmax=5)
```

Ornaments

```
ax.legend(…)
```

handles, labels, loc, title, frameon




```
ax.colorbar(…)
```

mappable, ax, cax, orientation

```
ax.annotate(…)
```

text, xy, xytext, xycoords, textcoords, arrowprops

Event handling

```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

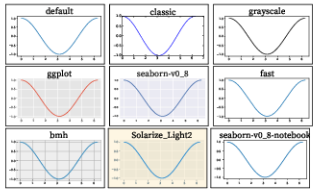
Animation

```
import matplotlib.animation as mpla

T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line, = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Styles

```
plt.style.use(style)
```



Quick reminder

```
ax.grid()
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(ticks, [labels])
ax.set_[xy]ticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

fig.suptitle(title)
fig.tight_layout()
plt.gcf(), plt.gca()
mpl.rc('axes', linewidth=1, …)
[fig|ax].patch.set_alpha(0)
text=r'$\frac{-e^{i\pi}}{2^n}$'
```

Keyboard shortcuts

ctrl + s — Save	ctrl + w — Close plot
r — Reset view	f — Fullscreen 0/1
f — View forward	b — View back
p — Pan view	o — Zoom to rect
x — X pan/zoom	y — Y pan/zoom
g — Minor grid 0/1	G — Major grid 0/1
l — X axis log/linear	L — Y axis log/linear

Ten simple rules

1. Know your audience
2. Identify your message
3. Adapt the figure
4. Captions are not optional
5. Do not trust the defaults
6. Use color effectively
7. Do not mislead the reader
8. Avoid "chartjunk"
9. Message trumps beauty
10. Get the right tool

1. **Gráficos Básicos**:
   - `plt.plot()`: Gráfico de linha.
   - `plt.bar()`: Gráfico de barras.
   - `plt.hist()`: Histograma.
   - `plt.scatter()`: Gráfico de dispersão.
   - `plt.pie()`: Gráfico de pizza.
2. **Customização**:
   - `plt.title()`: Define o título do gráfico.
   - `plt.xlabel()` / `plt.ylabel()`: Adiciona rótulos aos eixos.
   - `plt.legend()`: Adiciona legendas.
   - `plt.grid()`: Adiciona uma grade ao gráfico.
3. **Subplots**:
   - `plt.subplot()`: Divide o espaço da figura para múltiplos gráficos.
   - `plt.subplots()`: Mais flexível, retorna figuras e eixos.
4. **Estilização**:
   - Controle de cores, tipos de linhas e marcadores: `color`, `linestyle`, `marker`.
   - Paletas de cores personalizadas ou predefinidas.


seaborn

Principais Funções

1. **Gráficos de Distribuição**:
   - `sns.histplot()` : Histograma.
   - `sns.kdeplot()` : Densidade Kernel.
   - `sns.displot()` : Combinação de histograma e densidade.
2. **Relacionamentos**:
   - `sns.scatterplot()` : Gráfico de dispersão.
   - `sns.lineplot()` : Gráfico de linha.
   - `sns.regplot()` : Regressão com dispersão.
3. **Gráficos Categóricos**:
   - `sns.barplot()` : Barras com médias e desvios padrão.
   - `sns.boxplot()` : Box plot.
   - `sns.violinplot()` : Distribuições e estatísticas.
4. **Gráficos de Correlação**:
   - `sns.heatmap()` : Mapa de calor.
5. **Múltiplos Gráficos**:
   - `sns.pairplot()` : Gráficos de pares para múltiplas variáveis.

datacamp

Python For Data Science
Seaborn Cheat Sheet

Learn Seaborn online at www.DataCamp.com

Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot
5. Show your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips") #Step 1
>>> sns.set_style("whitegrid") #Step 2
>>> g = sns.lmplot(x="tip", #Step 3
                   y="total_bill",
                   data=tips,
                   aspect=2)
>>> g = (g.set_axis_labels("Tip","Total bill(USD)").
set(xlim=(0,10),ylim=(0,100)))
>>> plt.title("title") #Step 4
>>> plt.show(g) #Step 5
```

1 Data

Also see Lists, NumPy & Pandas

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({'x':np.arange(1,101),
                         'y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

2 Figure Aesthetics

Also see Matplotlib

```
>>> f, ax = plt.subplots(figsize=(5,6)) #Create a figure and one subplot
```

Seaborn styles

```
>>> sns.set() #(Re)set the seaborn default
>>> sns.set_style("whitegrid") #Set the matplotlib parameters
>>> sns.set_style("ticks", #Set the matplotlib parameters
                  {"xtick.major.size":8,
                   "ytick.major.size":8})
#Return a dict of params or use with with to temporarily set the style
>>> sns.axes_style("whitegrid")
```

3 Plotting With Seaborn

Axis Grids

```
>>> g = sns.FacetGrid(titanic, #Subplot grid for plotting conditional relationships
                       col="survived",
                       row="sex")
>>> g = g.map(plt.hist,"age")
>>> sns.factorplot(x="pclass", #Draw a categorical plot onto a Facetgrid
                   y="survived",
                   hue="sex",
                   data=titanic)
>>> sns.lmplot(x="sepal_width", #Plot data and regression model fits across a FacetGrid
               y="sepal_length",
               hue="species",
               data=iris)
>>> h = sns.PairGrid(iris) #Subplot grid for plotting pairwise relationships
>>> h = h.map(plt.scatter)
>>> sns.pairplot(iris) #Plot pairwise bivariate distributions
>>> i = sns.JointGrid(x="x", #Grid for bivariate plot with marginal univariate plots
                      y="y",
                      data=data)
>>> i = i.plot(sns.regplot,
               sns.distplot)
>>> sns.jointplot("sepal_length", #Plot bivariate distribution
                  "sepal_width",
                  data=iris,
                  kind='kde')
```

4 Further Customizations

Also see Matplotlib

Axisgrid Objects

```
>>> g.despine(left=True) #Remove left spine
>>> g.set_ylabels("Survived") #Set the labels of the y-axis
>>> g.set_xticklabels(rotation=45) #Set the tick labels for x
>>> g.set_axis_labels("Survived", #Set the axis labels
                      "Sex")
>>> h.set(xlim=(0,5), #Set the limit and ticks of the x-and y-axis
          ylim=(0,5),
          xticks=[0,2.5,5],
          yticks=[0,2.5,5])
```

Plot

```
>>> plt.title("A Title") #Add plot title
>>> plt.ylabel("Survived") #Adjust the label of the y-axis
>>> plt.xlabel("Sex") #Adjust the label of the x-axis
>>> plt.ylim(0,100) #Adjust the limits of the y-axis
>>> plt.xlim(0,10) #Adjust the limits of the x-axis
>>> plt.setp(ax,yticks=[0,5]) #Adjust a plot property
>>> plt.tight_layout() #Adjust subplot params
```

Context Functions

```
>>> sns.set_context("talk") #Set context to "talk"
>>> sns.set_context("notebook", #Set context to "notebook",
                    font_scale=1.5, #Scale font elements and
                    rc={"lines.linewidth":2.5}) #override param mapping
```

Color Palette

```
>>> sns.set_palette("husl",3) #Define the color palette
>>> sns.color_palette("husl") #Use with with to temporarily set palette
>>> flatui = ["#9b59b6","#3498db","#95a5a6","#e74c3c","#34495e","#2ecc71"]
>>> sns.set_palette(flatui) #Set your own color palette
```

Regression Plots

```
>>> sns.regplot(x="sepal_width", #Plot data and a linear regression model fit
                y="sepal_length",
                data=iris,
                ax=ax)
```

Distribution Plots

```
>>> plot = sns.distplot(data.y, #Plot univariate distribution
                        kde=False,
                        color="b")
```

Matrix Plots

```
>>> sns.heatmap(uniform_data,vmin=0,vmax=1) #Heatmap
```

Categorical Plots

Scatterplot

```
>>> sns.stripplot(x="species", #Scatterplot with one categorical variable
                  y="petal_length",
                  data=iris)
>>> sns.swarmplot(x="species", #Categorical scatterplot with non-overlapping points
                  y="petal_length",
                  data=iris)
```

Bar Chart

```
>>> sns.barplot(x="sex", #Show point estimates & confidence intervals with scatterplot glyphs
                y="survived",
                hue="class",
                data=titanic)
```

Count Plot

```
>>> sns.countplot(x="deck", #Show count of observations
                  data=titanic,
                  palette="Greens_d")
```

Point Plot

```
>>> sns.pointplot(x="class", #Show point estimates & confidence intervals as rectangular bars
                  y="survived",
                  hue="sex",
                  data=titanic,
                  palette={"male":"g",
                  "female":"m"},
                  markers=["^", "o"],
                  linestyles=["-","--"])
```

Boxplot

```
>>> sns.boxplot(x="alive", #Boxplot
                y="age",
                hue="adult_male",
                data=titanic)
>>> sns.boxplot(data=iris,orient="h") #Boxplot with wide-form data
```

Violinplot

```
>>> sns.violinplot(x="age", #Violin plot
                   y="sex",
                   hue="survived",
                   data=titanic)
```

5 Show or Save Plot

Also see Matplotlib

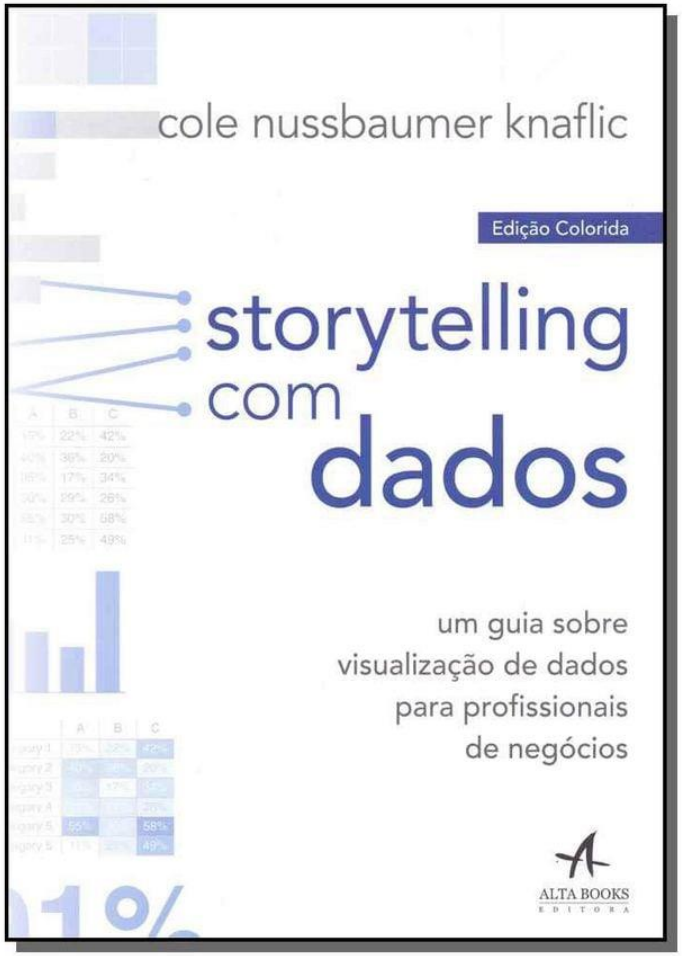
```
>>> plt.show() #Show the plot
>>> plt.savefig("foo.png") #Save the plot as a figure
>>> plt.savefig("foo.png", #Save transparent figure
                transparent=True)
```

> Close & Clear

Also see Matplotlib

```
>>> plt.cla() #Clear an axis
>>> plt.clf() #Clear an entire figure
>>> plt.close() #Close a window
```

Learn Data Skills Online at www.DataCamp.com


cole nussbaumer knaflic
Edição Colorida
storytelling com dados
um guia sobre visualização de dados para profissionais de negócios
ALTA BOOKS
EDITORA