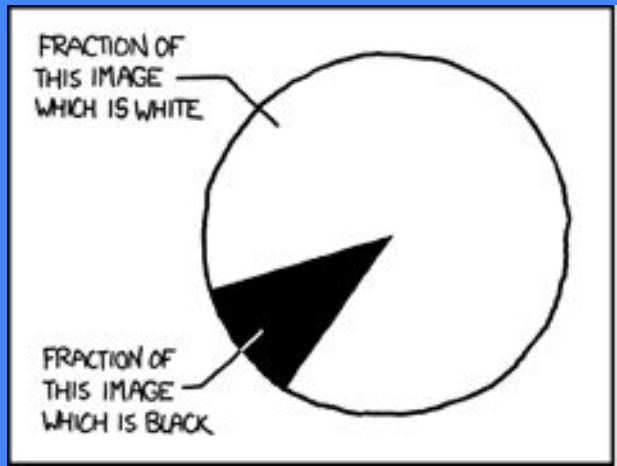
Programación para el análisis de datos

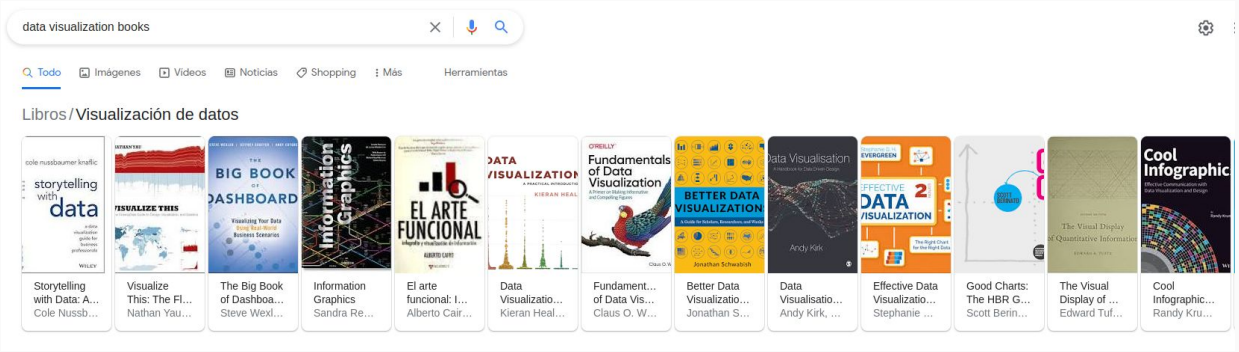
Federico Pousa
fpousa@udesa.edu.ar

Hoy:
Gráficos



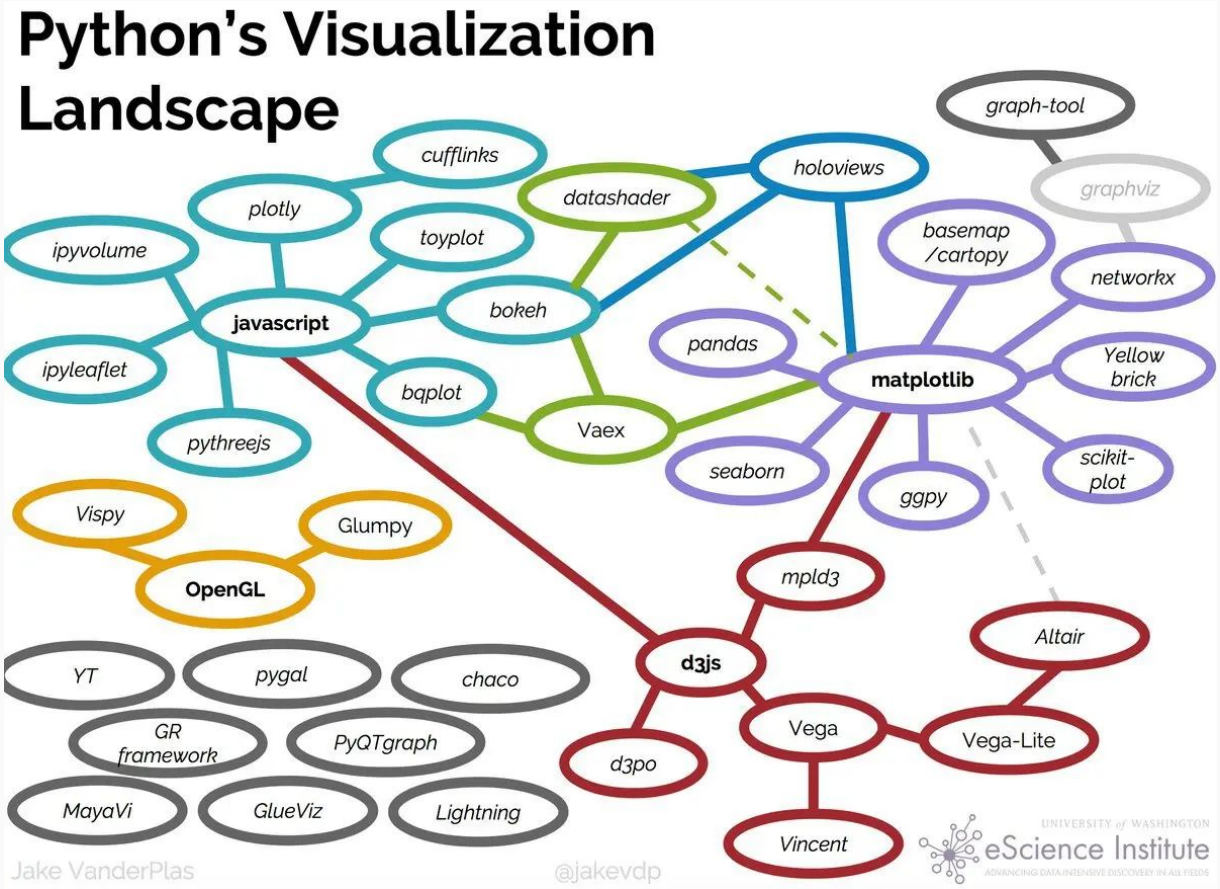
Graficando

- Son una herramienta muy poderosa para entender nuestros datos y formular hipótesis.
- Nos ayudan a contar una historia.
- Más allá del conocimiento “tecnológico” para graficar, hay un montón de conocimiento extra que determinan el éxito o fracaso de una visualización:
  - No cualquier gráfico da lo mismo.
  - No cualquier color da lo mismo.
  - No cualquier recorte da lo mismo.
  - Etc, etc, etc.



Más en: **Visualización y comunicación...**

A no perderse



Matplotlib

Es una librería para ploteos (mayormente) en 2D (https://matplotlib.org/)

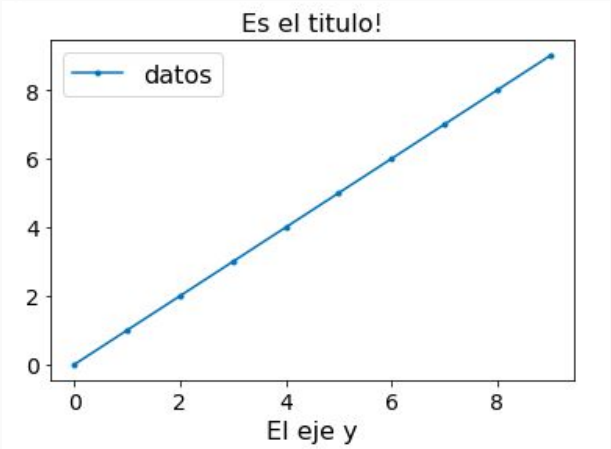
¿Cómo instalarla?

```
pip install matplotlib
```

Ejemplo:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.plot(range(10),'.-', label='datos')
plt.title('Es el titulo!',size=16)
plt.xlabel('El eje x',size=16)
plt.xlabel('El eje y',size=16)
plt.legend(fontsize=16)
plt.xticks(size=14)
plt.yticks(size=14)
plt.show()
```



Matplotlib: Conceptos

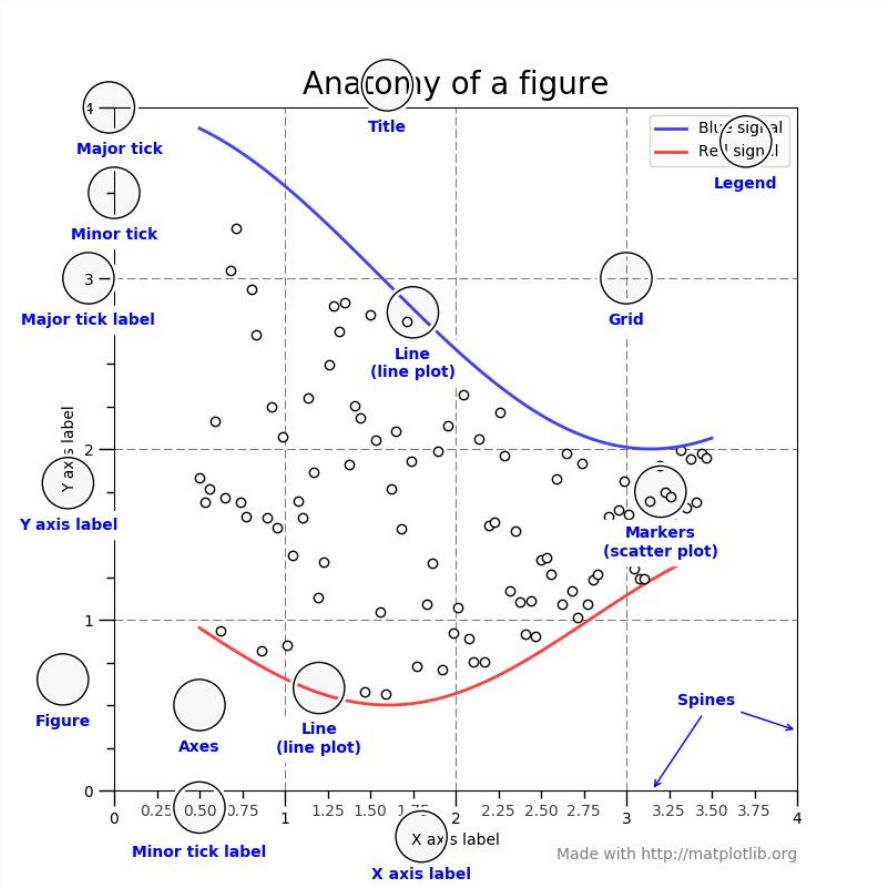
- **Figure:** Es todo el área donde se va a realizar el “dibujo” final. Puede contener 1 o más *Axes*

- **Axes:** Es un gráfico en sí. Pertenece a una sola Figure, pero comparte potencialmente el espacio con otros Axes en esa Figure. Contiene 2 o 3 *Axis*.

- **Axis:** Son los ejes del gráfico, contienen toda la información del eje, cada cuanto hay una label, cómo se escribe, etc.

Referencia: https://matplotlib.org/stable/tutorials/introductory/usage.html

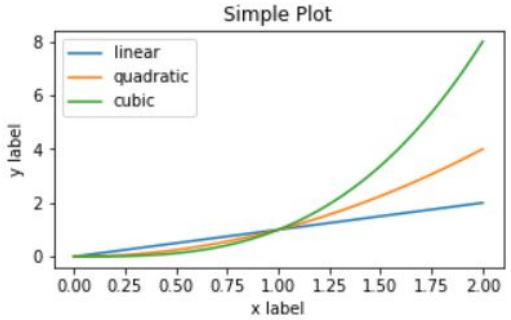
Matplotlib: Conceptos



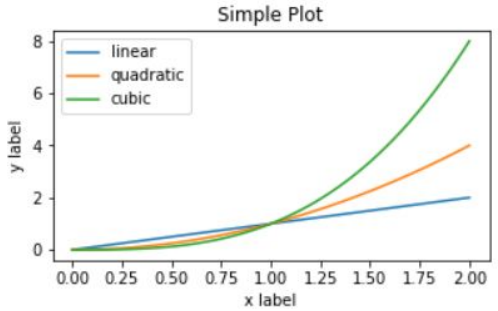
Formas de interactuar

Hay dos maneras de interactuar con la librería. Queda a gusto del consumidor qué manera a utilizar (y a gusto del que escribió el código original que nos estamos copiando)

In [2]:
```
x = np.linspace(0, 2, 100)  # Sample data.

fig, ax = plt.subplots(figsize=(5, 2.7))
ax.plot(x, x, label='linear')  # Plot some data or
ax.plot(x, x**2, label='quadratic')  # Plot more d
ax.plot(x, x**3, label='cubic')  # ... and some mo
ax.set_xlabel('x label')  # Add an x-label to the
ax.set_ylabel('y label')  # Add a y-label to the a
ax.set_title("Simple Plot")  # Add a title to the
ax.legend();  # Add a legend.
```



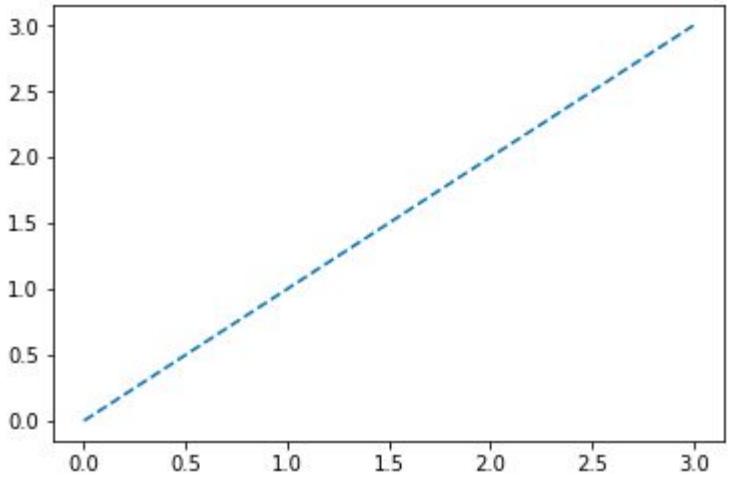
In [3]:
```
x = np.linspace(0, 2, 100)  # Sample data.

plt.figure(figsize=(5, 2.7))
plt.plot(x, x, label='linear')  # Plot some data
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend();
```



Matplotlib: Ejemplos

```
fig = plt.Figure()
ax = plt.plot([0,1,2,3],'--')
print('Tamano de ax',len(ax))
print('Tipo de ax[0]',type(ax[0]))
```

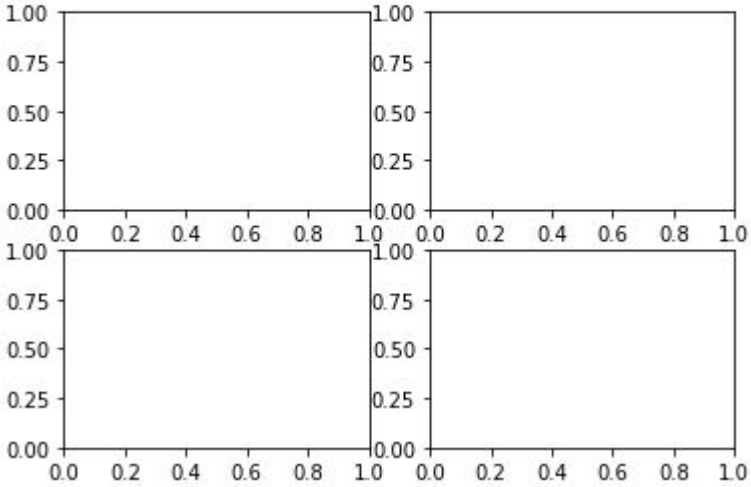
```
Tamano de ax 1
Tipo de ax[0] <class 'matplotlib.lines.Line2D'>
```



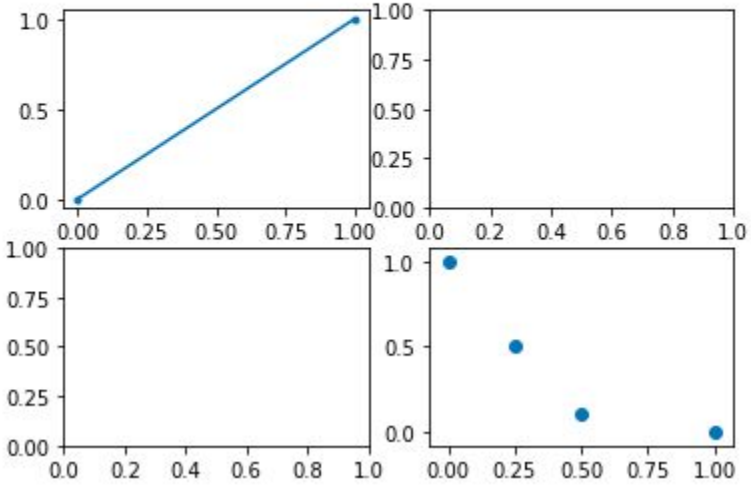
Matplotlib: Subplots

```
fig, axes = plt.subplots(2,2)
axes.shape
```

```
(2, 2)
```



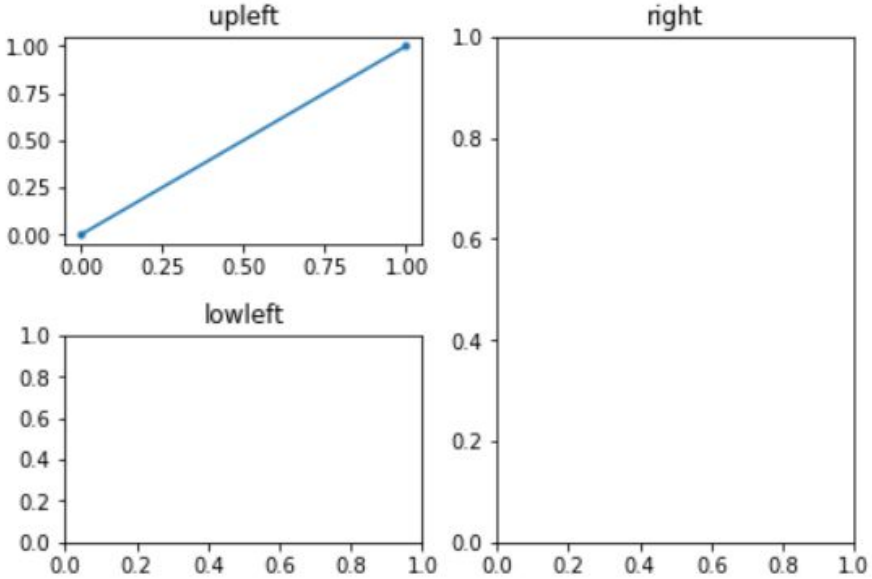
```
fig, axes = plt.subplots(2,2)
axes[0,0].plot([0,1],'.-')
axes[1,1].scatter([1,0.25,0.5,0],[0,0.5,0.1,1])
plt.show()
```



Matplotlib: Subplots

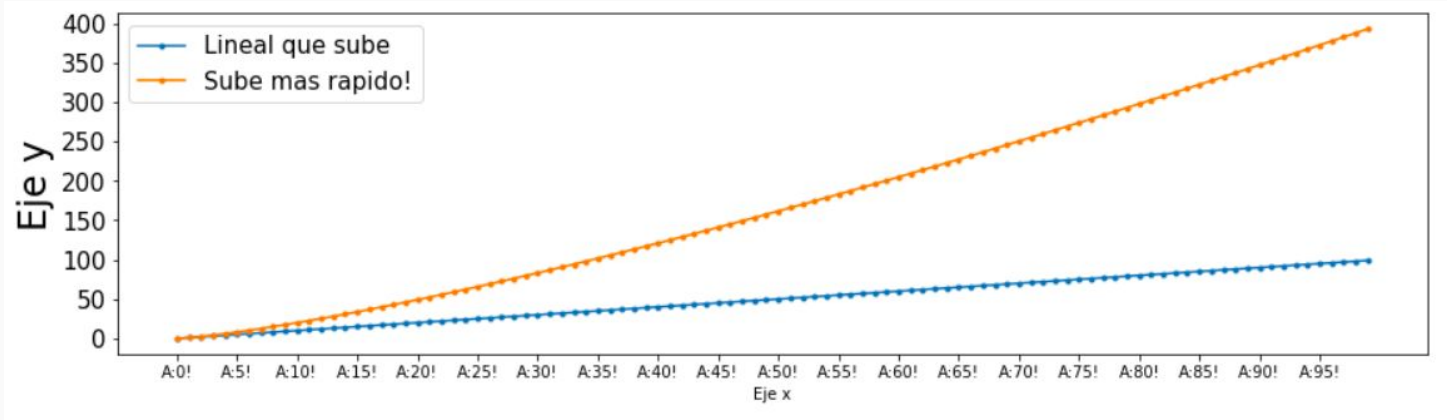
In [5]:
```
fig, axes = plt.subplot_mosaic([['upleft', 'right'],
                                ['lowleft', 'right']], layout='constrained')
axes['upleft'].set_title('upleft')
axes['upleft'].plot([0,1], '.-')
axes['lowleft'].set_title('lowleft')
axes['right'].set_title('right');
```

Evita que se superpongan los titles entre subplots



Matplotlib: Label, legend, etc

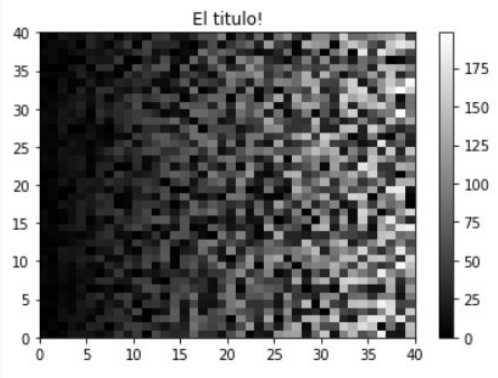
```
fig ,ax = plt.subplots(1,1)
fig.set_size_inches([15,4]) # seteo el tamano del figure

ax.plot(range(100),'.-',label='Lineal que sube')
ax.plot( [x**1.3 for x in range(100)],'.-',label='Sube mas rapido!')

# Cambio el tamano de la letra del legend
plt.legend(fontsize=15)

# modifico los ticks
plt.xticks(range(0,100,5),[ 'A:'+str(e)+'!' for e in range(0,100,5)])

# modifico la letra de los ticks y
plt.yticks(fontsize=15)

# labels
plt.xlabel('Eje x')
plt.ylabel('Eje y',size=24)

plt.show()
```



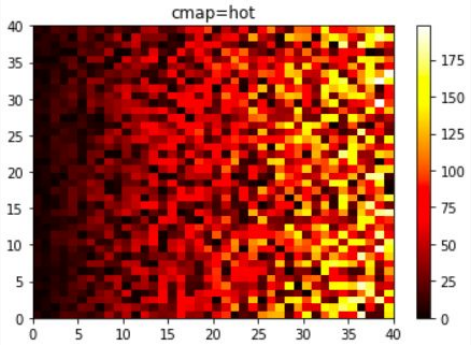
Matplotlib: Matrices

```
import numpy as np
m = np.concatenate([np.random.randint(0,10+10*i,[40,2]) for i in range(20)],axis=1)
```

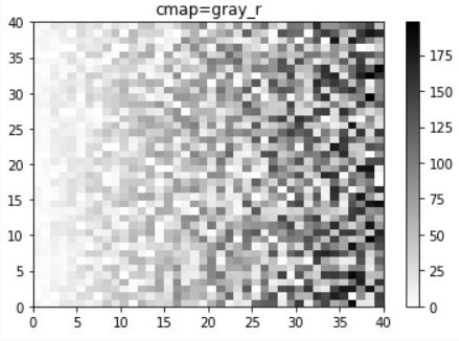
```
plt.pcolor(m,cmap='gray')
plt.colorbar()
plt.title('El titulo!')
```




```
fig, ax = plt.subplots(1,1)
grafico = ax.pcolor(m,cmap='hot')
fig.colorbar(grafico)
ax.set_title('cmap=hot')
```

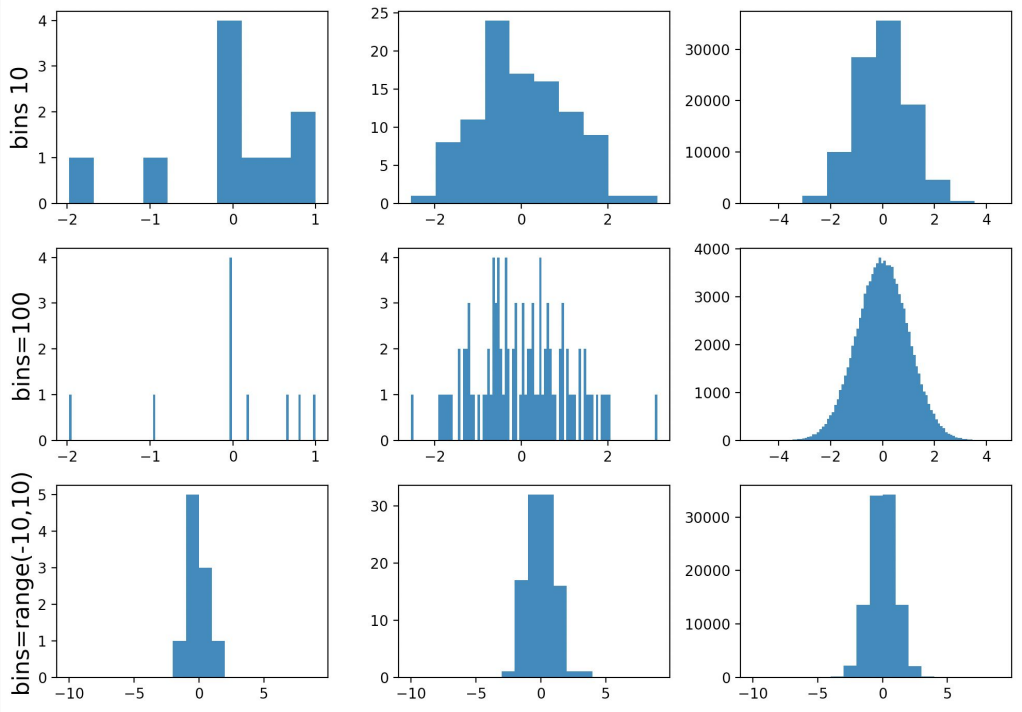



```
plt.pcolor(m,cmap=gray_r)
plt.colorbar()
plt.title(cmap=gray_r)
```



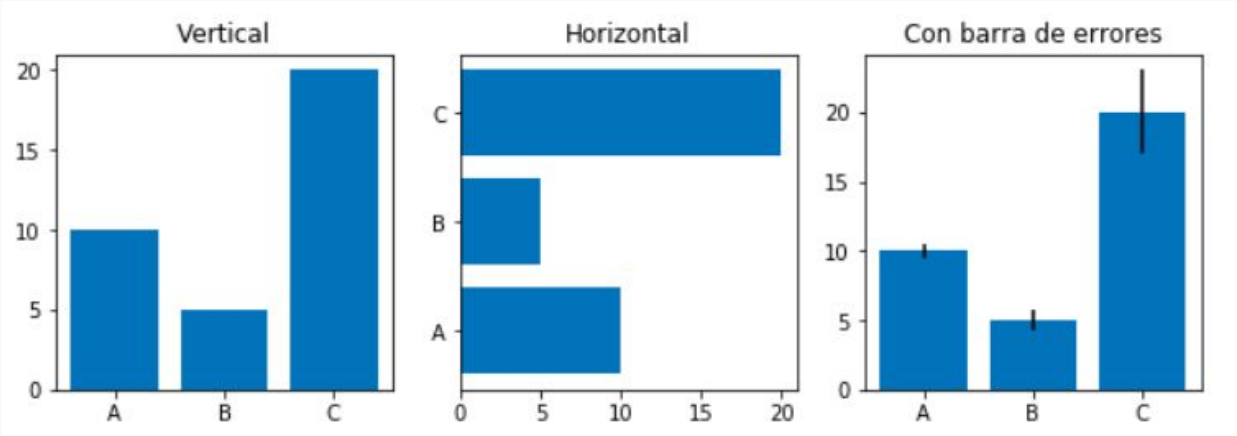
Matplotlib: Histograma

```
fig , axes = plt.subplots(3,3)

fig.set_size_inches([10,7])
dist_10 = np.random.normal(size=10)
dist_100 = np.random.normal(size=100)
dist_100000 = np.random.normal(size=100000)


axes[0,0].hist(dist_10,bins=10)
axes[0,0].set_ylabel('bins 10',fontsize=16)
axes[0,1].hist(dist_100,bins=10)
axes[0,2].hist(dist_100000,bins=10)

axes[1,0].hist(dist_10,bins=100)
axes[1,0].set_ylabel('bins=100',fontsize=16)
axes[1,1].hist(dist_100,bins=100)
axes[1,2].hist(dist_100000,bins=100)

axes[2,0].hist(dist_10,bins=range(-10,10))
axes[2,0].set_ylabel('bins=range(-10,10)',fontsize=16)
axes[2,1].hist(dist_100,bins=range(-10,10))
axes[2,2].hist(dist_100000,bins=range(-10,10))

fig.tight_layout()
```



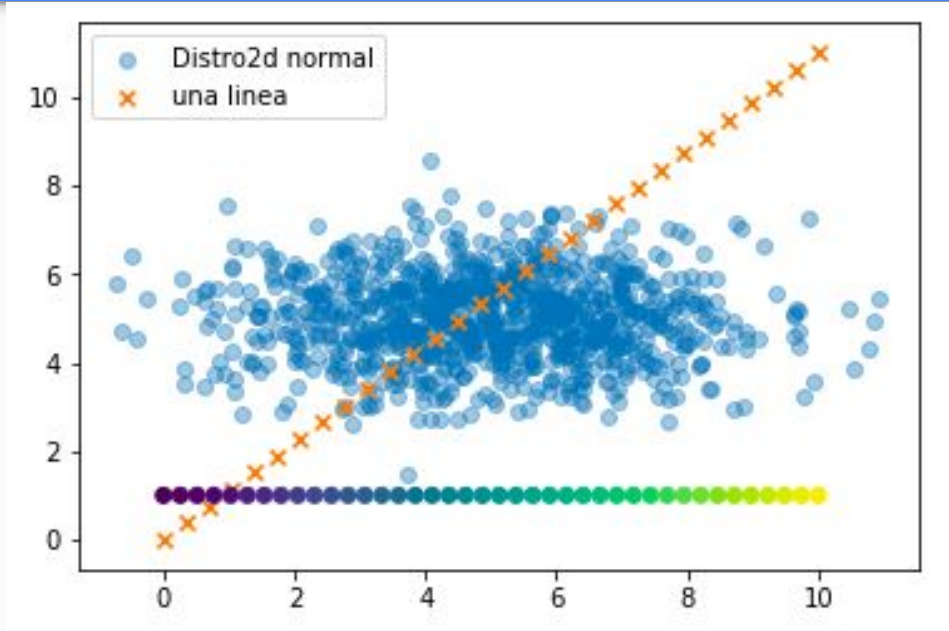
Matplotlib: Barras

```
fig , axes = plt.subplots(1,3)

fig.set_size_inches([10,3])

axes[0].bar(['A','B','C'],[10,5,20])
axes[0].set_title('Vertical')

axes[1].barh(['A','B','C'],[10,5,20])
axes[1].set_title('Horizontal')

axes[2].bar(['A','B','C'],[10,5,20],yerr=[0.5,0.8,3])
axes[2].set_title('Con barra de errores')
```



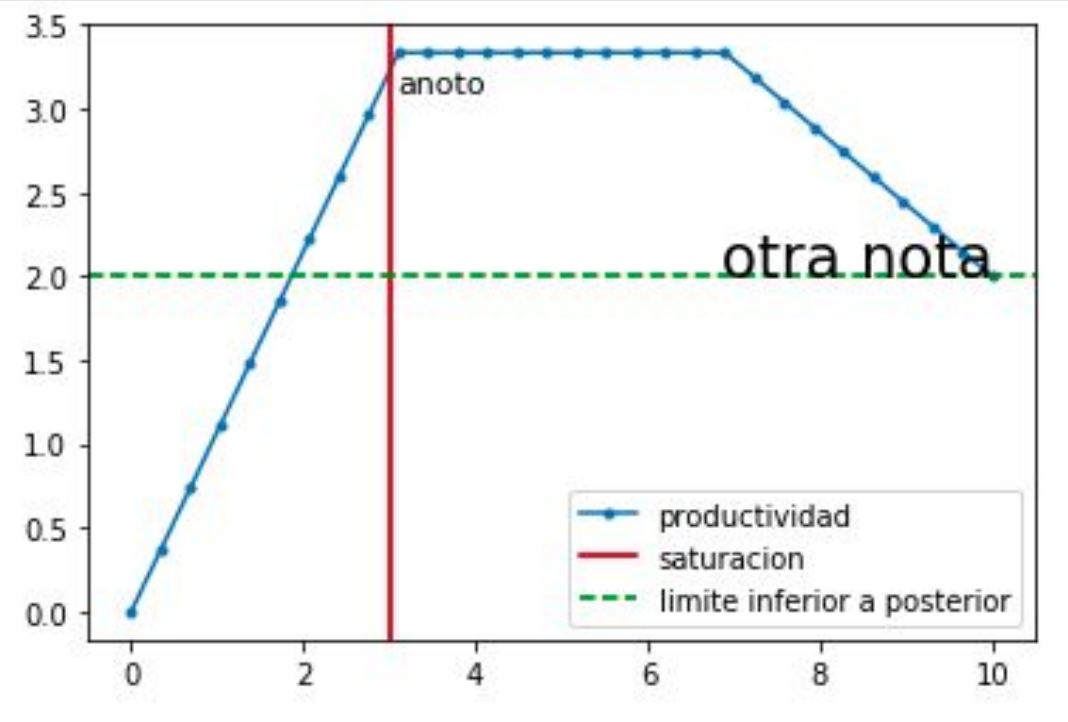
Matplotlib: Scatter

```
# Creo una distribucion normal en 2d
# centrada en (5,5) pero con desvio
# horizontal mas grande
mx = np.random.normal(5,2 , size=1000)
my = np.random.normal(5,1, size=1000)

# una recta
lx = np.linspace(0,10,30)
ly = lx*1.1


plt.scatter(mx,my,alpha=0.4,label='Distro2d normal')

# Puedo determinar el tipo de marker
plt.scatter(lx,ly,marker='x',label='una linea')

# Le paso en el campo color un array de valores
# podria pasarle tambien un color, por ejemplo: "red"
plt.scatter(np.linspace(0,10,40),[1]*40,c=np.linspace(0,10,40))
```



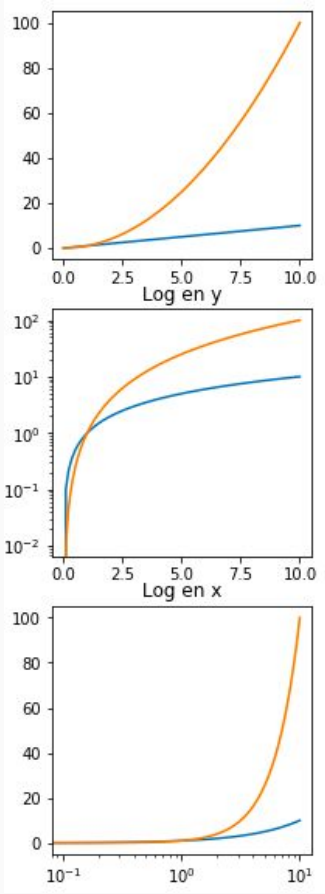
Matplotlib: Anotaciones y líneas

```
lx = np.linspace(0,10,30)
ly1 = np.linspace(0,10/3,10)
ly2 = np.ones(10)*10/3
ly3 = np.linspace(10/3,2,10)
ly = np.concatenate([ly1,ly2,ly3])

plt.plot(lx,ly,'.-',label='productividad')
plt.axvline(3,color='red',label='saturacion')

plt.axhline(2,linestyle='--',color='green',label='limite inferior a posterior')
plt.text(3.1,10/3-0.1,'anoto',fontsize=11,verticalalignment='top')
plt.text(10,2,'otra nota',fontsize=21,horizontalalignment='right')
plt.legend()
```



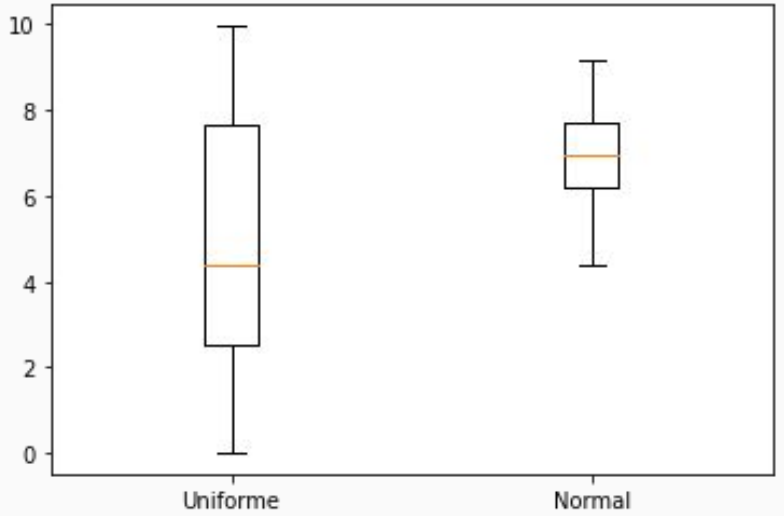
Matplotlib: Cambio escala de ejes

```
x = np.linspace(0,10,100)
y1 = np.linspace(0,10,100)
y2 = np.linspace(0,10,100)**2

fig, axes = plt.subplots(3,1)
fig.set_size_inches((10,3))
axes[0].plot(x,y1)
axes[0].plot(x,y2)
axes[0].set_title('')

axes[1].plot(x,y1)
axes[1].plot(x,y2)
axes[1].set_yscale("log")
axes[1].set_title('Log en y')


axes[2].plot(x,y1)
axes[2].plot(x,y2)
axes[2].set_xscale("log")
axes[2].set_title('Log en x')
plt.show()
```



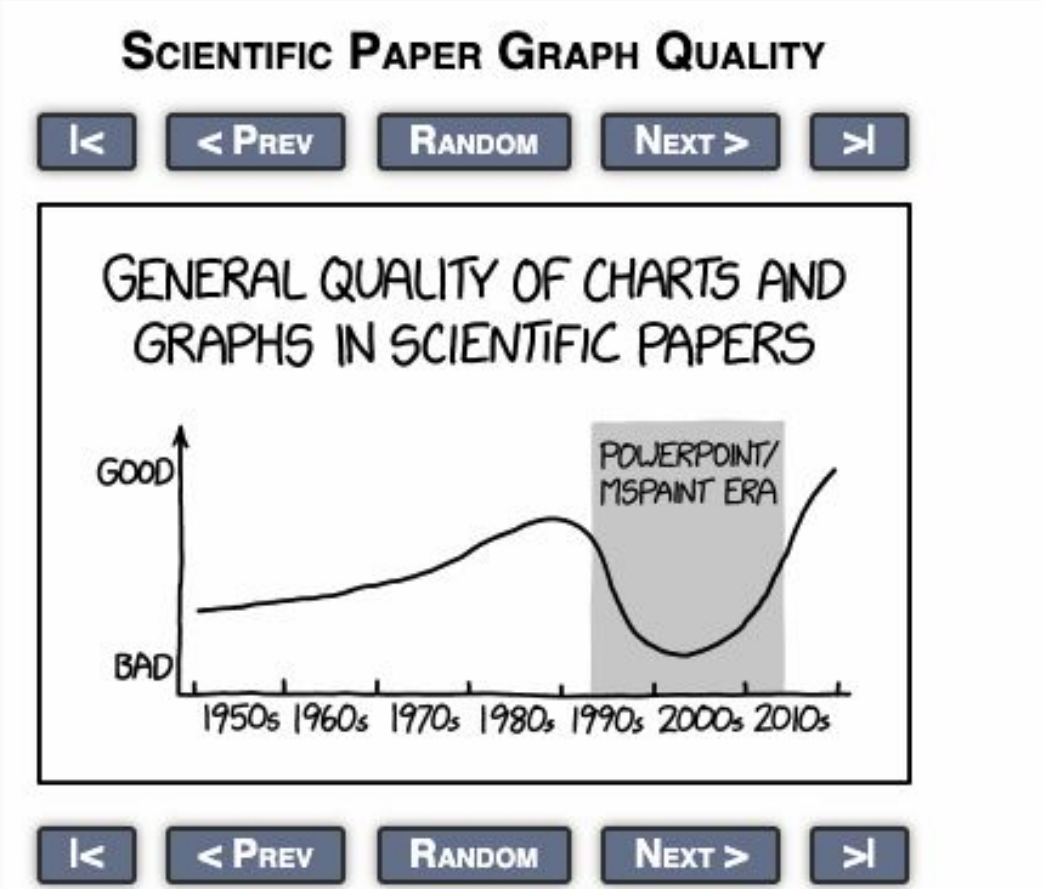
Matplotlib: Boxplots

```
u = 5+(np.random.rand(100)-0.5)*10
n = np.random.normal(7,1,100)
plt.boxplot([u,n])
plt.xticks([1,2],['Uniforme','Normal'])
```



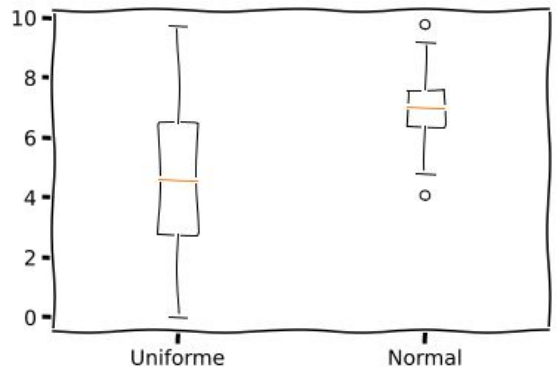
Matplotlib: Bonus track XKCD

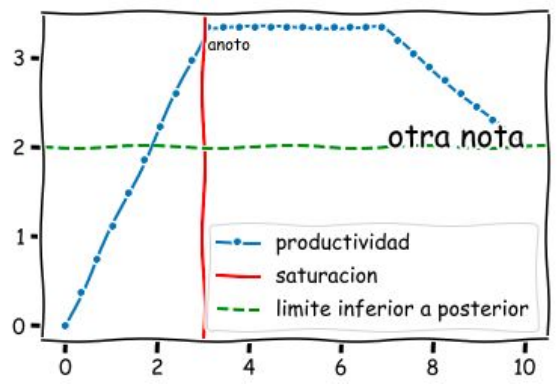
https://xkcd.com/1945/

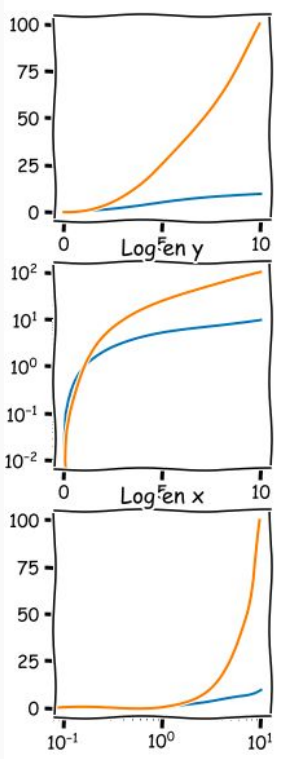


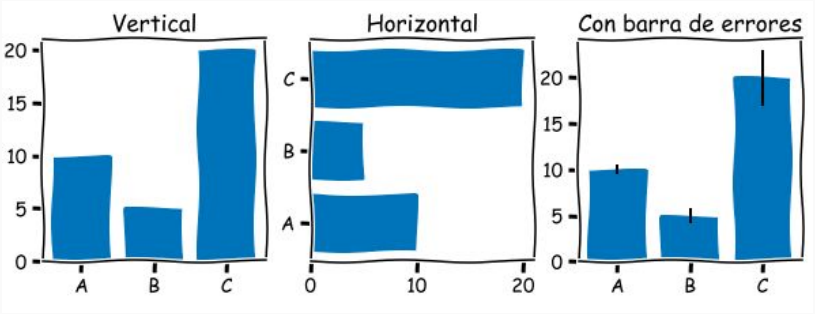
Matplotlib: Bonus track XKCD

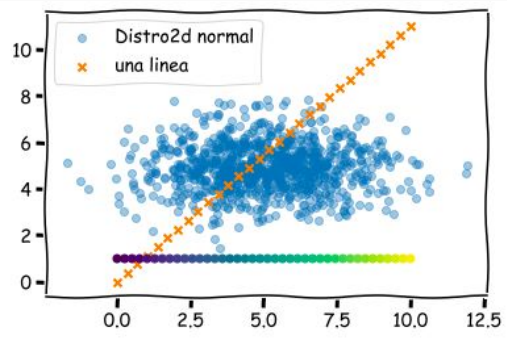
`plt.xkcd()` **Referencia:** https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xkcd.html











Matplotlib: Pandas

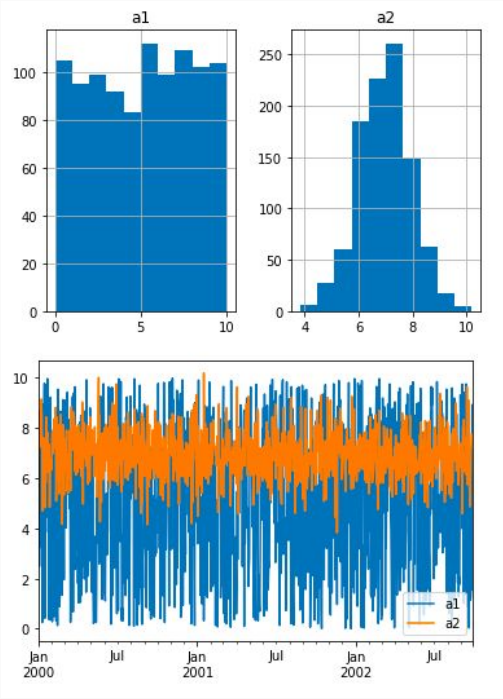
Los objetos de pandas son ploteables:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime

u = 5+(np.random.rand(1000)-0.5)*10
n = np.random.normal(7,1,1000)

# un rango de mil dias
dias = pd.date_range(start='2000-01-01',periods=1000)

df = pd.DataFrame(np.array([u,n]).T,
     columns=['a1','a2'],
     index=dias)
df.head(10)
```

	a1	a2
2000-01-01	9.935027	6.664970
2000-01-02	6.882377	6.438621
2000-01-03	4.127945	7.345508
2000-01-04	4.740101	6.725254
2000-01-05	3.100711	7.871478
2000-01-06	6.316484	5.787593
2000-01-07	2.114135	7.849160
2000-01-08	6.900114	7.948216
2000-01-09	8.685017	8.563538
2000-01-10	2.535088	6.005147

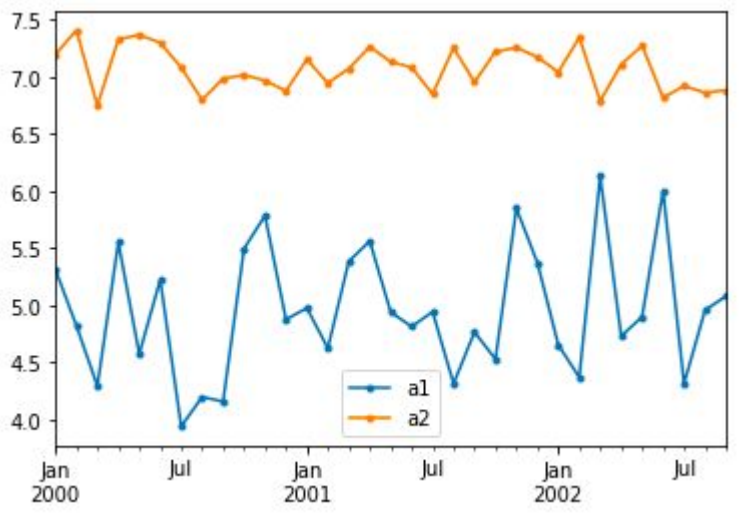
Matplotlib: Pandas

```
# ploteo los histogramas
df.hist()

# ploteo como lineas
df.plot()
```



Matplotlib: Pandas

```
# resampleo por mes, toma la media y ploteo
df.resample('m').mean().plot(style='.-')
```



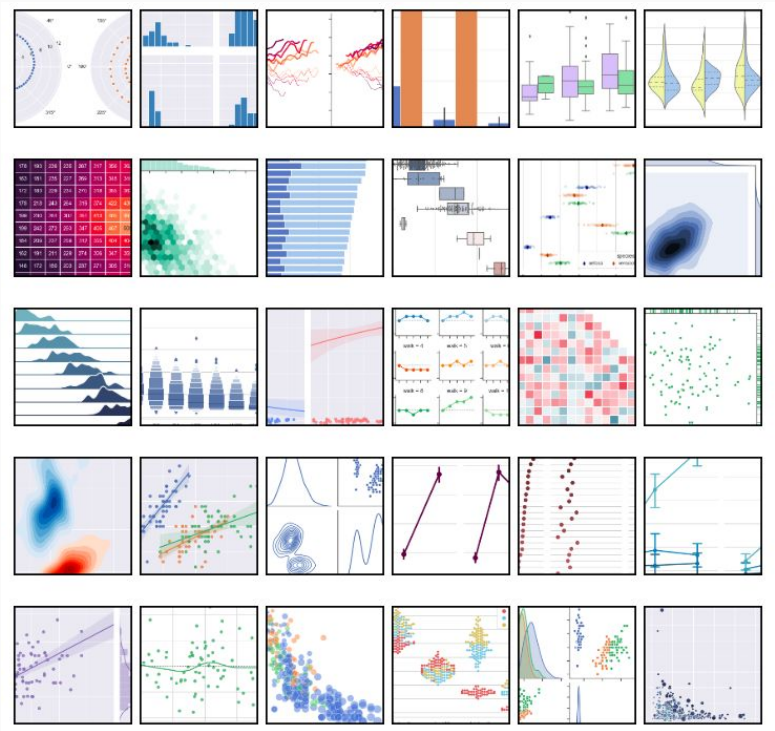
Ref: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html

Matplotlib: Gallery

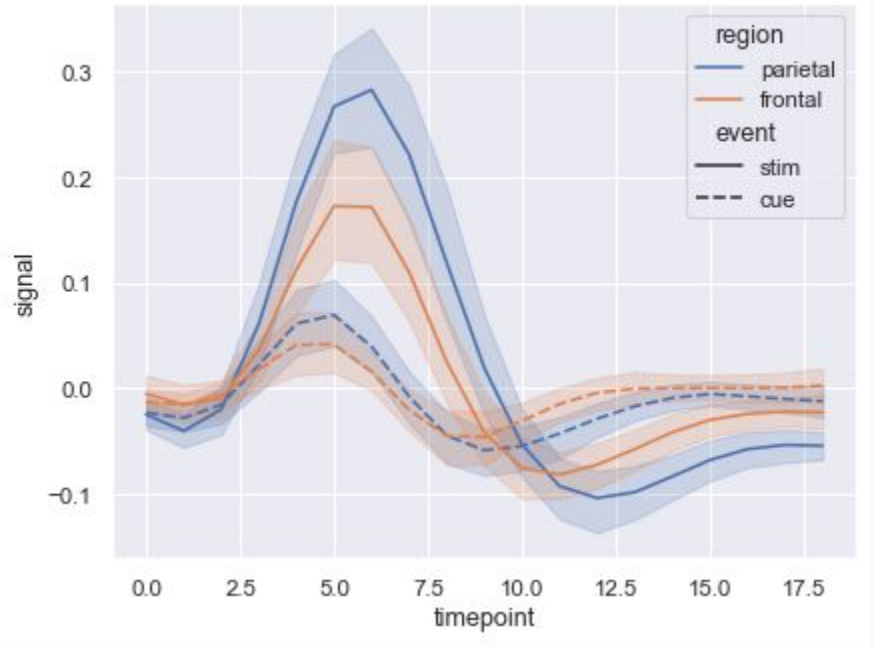
https://matplotlib.org/stable/gallery/

Seaborn: Matplotlib con esteroides

https://seaborn.pydata.org/introduction.html



LinePlot




```python
import seaborn as sns
sns.set_theme(style="darkgrid")

# Load an example dataset with long-form data
fmri = sns.load_dataset("fmri")

# Plot the responses for different events and regions
sns.lineplot(x="timepoint", y="signal",
             hue="region", style="event",
             data=fmri)
```

Error

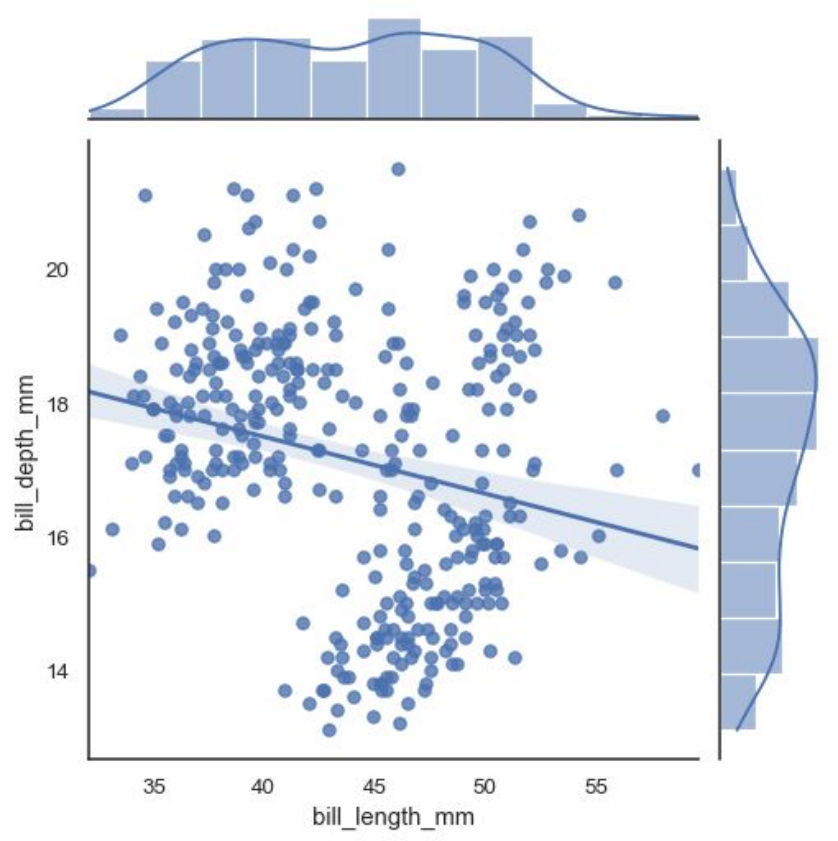
El gráfico anterior tiene información de márgenes de error porque nuestra información no era un único punto por valor de “x”.

```
In [14]: fmri
```

Out[14]:

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970
...	...	...	...	...	...
1059	s0	8	cue	frontal	0.018165
1060	s13	7	cue	frontal	-0.029130
1061	s12	7	cue	frontal	-0.004939
1062	s11	7	cue	frontal	-0.025367
1063	s0	0	cue	parietal	-0.006899

JointPlot




```
sns.jointplot(data=penguins,
x="bill_length_mm",
y="bill_depth_mm", kind="reg")
```

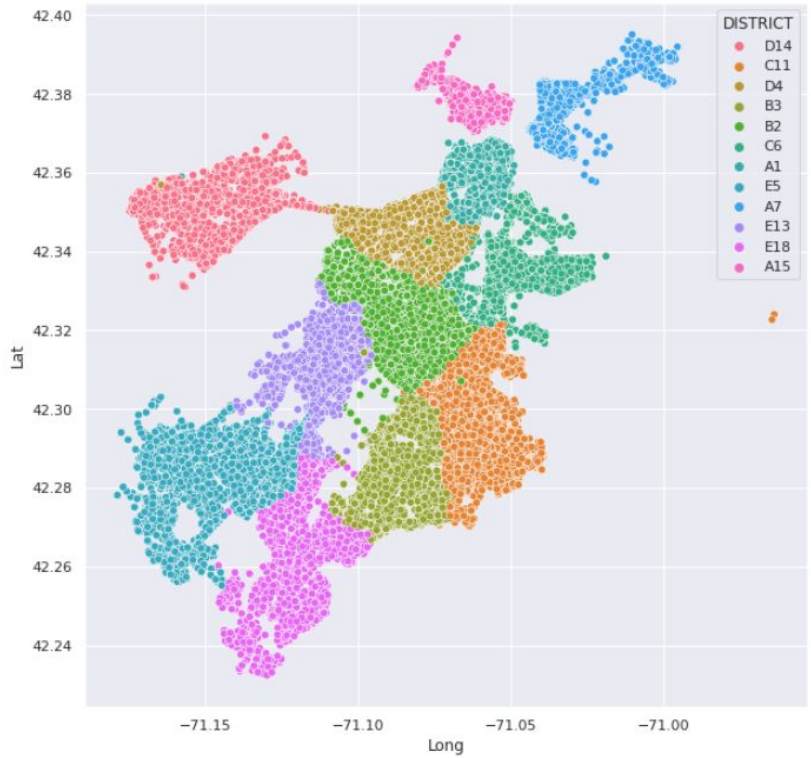
HeatMap


```
sns.heatmap(flights, annot=True,

fmt="d", linewidths=.5, ax=ax)
```

Integrando

```
In [16]: df = pd.read_csv("crime.csv", encoding="ISO-8859-1")
         fig, axs = plt.subplots(1, 1, figsize=(10, 10))
         sns.scatterplot(data=df[df['Long'] < -10], x='Long', y='Lat', hue='DISTRICT')
```

```
Out[16]: <AxesSubplot:xlabel='Long', ylabel='Lat'>
```



Integrando

In [17]:
```
heatmap = df.groupby([
    'DAY_OF_WEEK', 'HOUR'
])['INCIDENT_NUMBER'].count().reset_index()
heatmap.head()
```

Out[17]:

	DAY_OF_WEEK	HOUR	INCIDENT_NUMBER
0	Friday	0	2161
1	Friday	1	1275
2	Friday	2	952
3	Friday	3	532
4	Friday	4	441

In [18]:
```
heatmap = heatmap.pivot(columns='HOUR', index='DAY_OF_WEEK', values='INCIDENT_NUMBER')
```

In [19]:
```
fig, axs = plt.subplots(1, 1, figsize=(20, 5))
sns.heatmap(heatmap, annot=True, ax=axs, fmt=".0f")
```

Out[19]: <AxesSubplot:xlabel='HOUR', ylabel='DAY_OF_WEEK'>

BonusTrack: GeoPandas

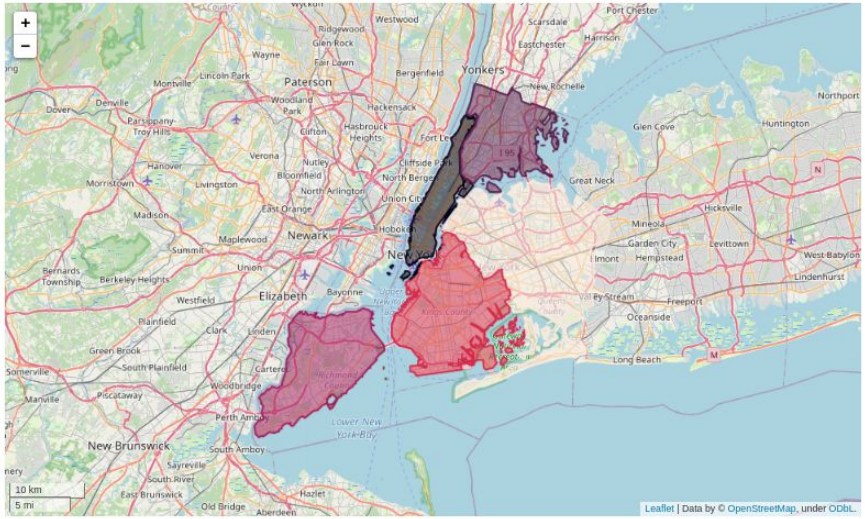
In [20]:
```
import geopandas

path_to_data = geopandas.datasets.get_path("nybb")
gdf = geopandas.read_file(path_to_data)
gdf = gdf.set_index("BoroName")
gdf["area"] = gdf.area

gdf
```

Out[20]:

BoroName	BoroCode	Shape_Leng	Shape_Area	geometry	area
Staten Island	5	330470.010332	1.623820e+09	MULTIPOLYGON (((970217.022 145643.332, 970227....	1.623822e+09
Queens	4	896344.047763	3.045213e+09	MULTIPOLYGON (((1029606.077 156073.814, 102957...	3.045214e+09
Brooklyn	3	741080.523166	1.937479e+09	MULTIPOLYGON (((1021176.479 151374.797, 102100...	1.937478e+09
Manhattan	1	359299.096471	6.364715e+08	MULTIPOLYGON (((981219.056 188655.316, 980940....	6.364712e+08
Bronx	2	464392.991824	1.186925e+09	MULTIPOLYGON (((1012821.806 229228.265, 101278...	1.186926e+09

In [21]:
```
gdf.explore("area", legend=False)
```

Out[21]:

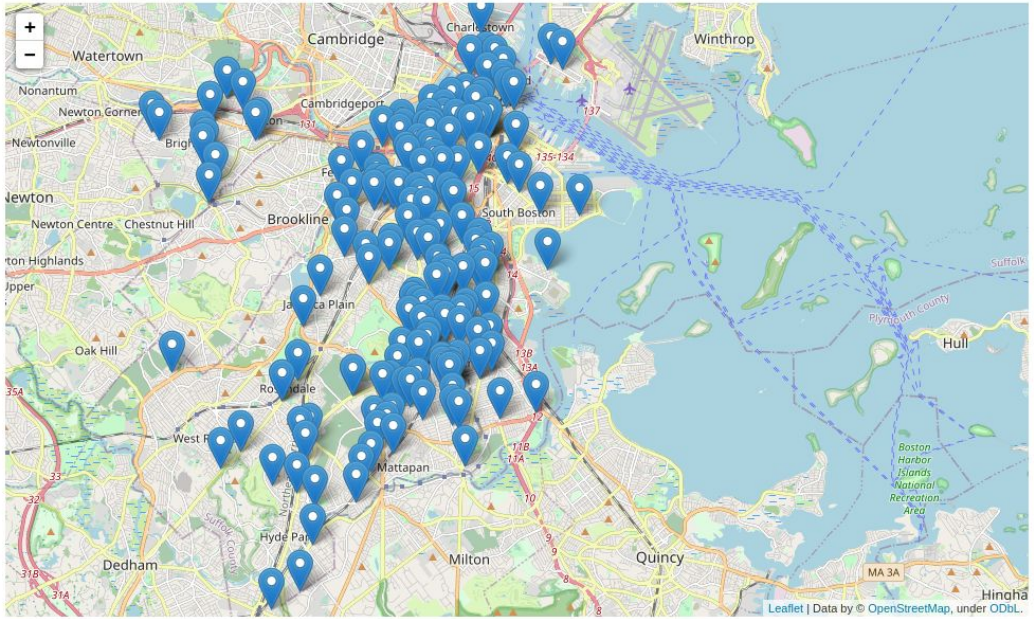


BonusTrack: Folium

In [22]:
```
df = pd.read_csv("crime.csv", encoding="ISO-8859-1")
df = df[df['Lat'].notna()]
df = df[df['Long'].notna()]
df = df.head(200)
```

In [23]:
```
import folium
locations = df[['Lat', 'Long']]
locationlist = locations.values.tolist()
map = folium.Map(location=[42.31, -71.05], zoom_start=12)
for point in range(0, len(locationlist)):
    folium.Marker(locationlist[point]).add_to(map)
map
```

Out[23]:



Guia de ejercicios

https://drive.google.com/file/d/19AK2Zn5Y9oCXm6ely7luFFvLq48xKiu_/view?usp=sharing