Python para Análisis de datos: Introducción

Sesión 6

Jesús Fernández (fernandez.cuesta@gmail.com)

25 Octubre 2018

Visualización de datos

Diferentes librerías para visualización de datos

- Matplotlib
  - standard de-facto
  - hacer fácil tareas sencillas y posible tareas complejas
  - versátil, madura y extensa
  - integrado con pandas (<100%)
  - acepta paquetes de terceros como **extensiones**
- Algunas extensiones de Matplotlib:
  - Seaborn
    - funciona sobre `matplotlib`
    - enfocado a análisis estadístico
  - Cartopy, folium
    - visualización de datos en mapas
- Bokeh
  - enfocado a gráficos interactivos

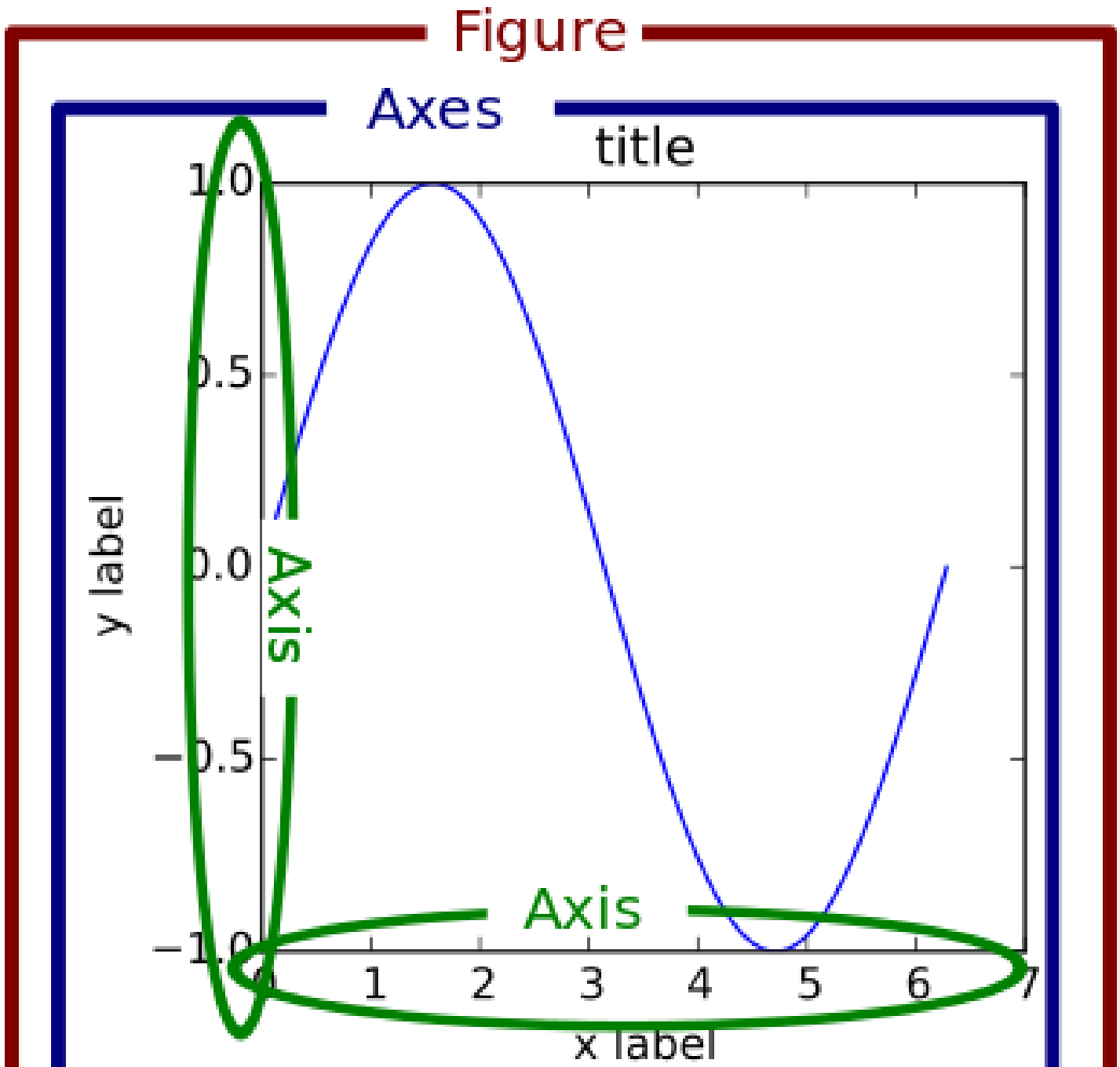
Matplotlib

`pyplot`: módulo con interfaz parecida a MATLAB

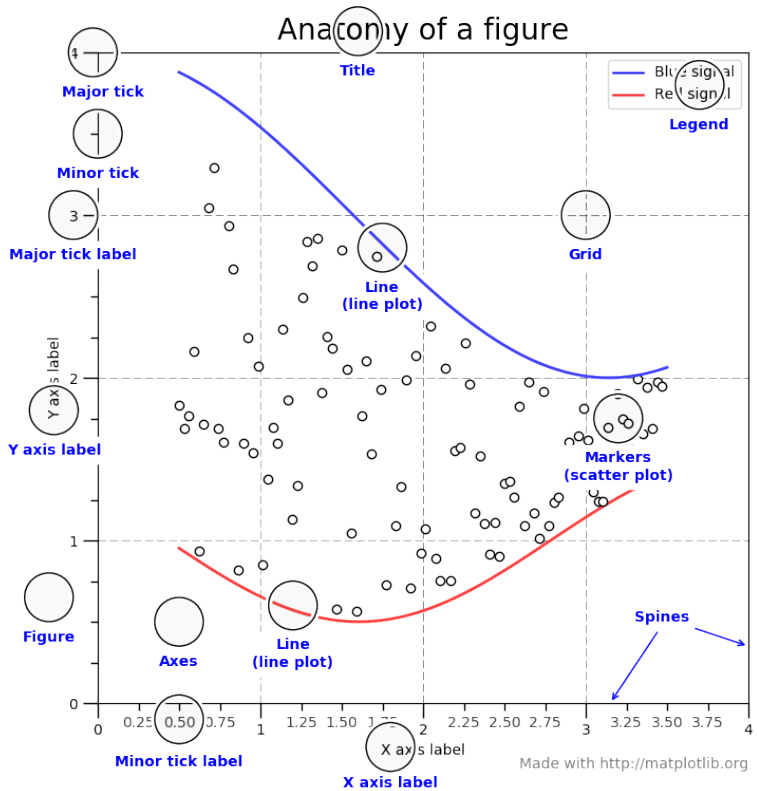
```
import matplotlib.pyplot as plt
# from matplotlib import pyplot as plt
```

Importar las librerías necesarias

Partes de una figura



Partes de una figura (II)



Los componentes más importantes son:

- Figura
- Ejes (axes)
  - región de la figura donde se visualizarán gráficos
- Eje (axis)
  - eje de coordenadas
- Artist
  - cualquier elemento (líneas, leyenda, etc., incluidos ejes)

Como regla general seguiremos los siguientes pasos:

1. Crear figura
2. Obtener ejes
3. Dibujar sobre los ejes

Nota: 1 y 2 se pueden combinar en un mismo comando

```
N = 1000
np.random.seed(2983)  # reproducibilidad

fig = plt.figure()  # crea figura
ax = plt.subplot(1, 1, 1)  # crea ejes
t = pd.date_range('1/1/2018', periods=365)
y = np.random.randn(N).cumsum()
ax.plot(t, y)  # dibuja sobre los ejes
```

De forma simplificada, los ejes y la figura se crean simultáneamente:

```
# fig = plt.figure()  # crea figura
# ax = plt.subplot(1, 1, 1)  # crea ejes
(fix, ax) = plt.subplots(1, 1)  # (n_rows, n_cols)
t = pd.date_range('1/1/2018', periods=365)
y = np.random.randn(N).cumsum()
ax.plot(t, y)  # dibuja sobre los ejes
```

Importante

En un cuaderno los gráficos aparecerán automáticamente si la primera línea es:

```
%matplotlib inline
```

De lo contrario necesitaremos ejecutar:

```
# [...]
ax.plot(t, y, 'g.-')
plt.show()  # muestra el gráfico
```

`...` para mostrar cada gráfico

matplotlib acepta los siguientes tipos de datos:

- Listas
- Diccionarios
- `np.array`
- `pandas.DataFrame`

El tipo de datos nativo es `np.array`.

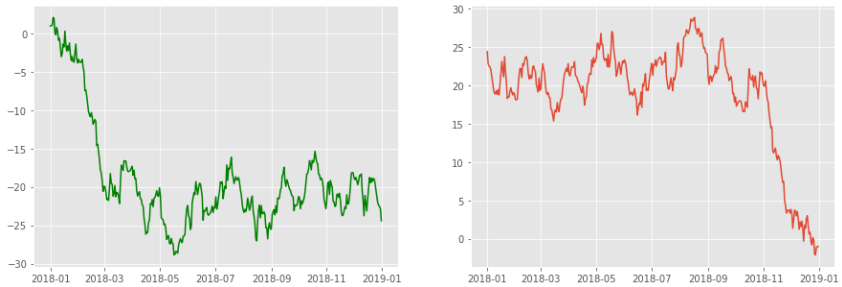
El resto puede -o no- funcionar.

`plt.subplots()`

Para crear una tupla (figura, ejes) usaremos:

- `plt.subplots()`
  - sin argumentos: crea una figura con 1 area de dibujo
  - (`nrows`, `ncols`): crea una figura con `nrows*ncols` areas
  - devuelve la figura y todos los ejes

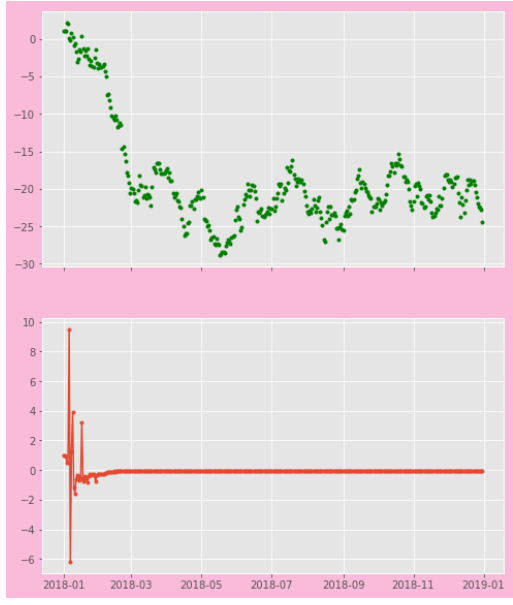
```
fig, (eje1, eje2) = plt.subplo

eje1.plot(t, y, 'g')
eje2.plot(t, -y[::-1])
```



`plt.subplots()`

Para crear una tupla (figura, ejes) usaremos:

- `plt.subplots()`
  - sin argumentos: crea una figura con 1 area de dibujo
  - (`nrows`, `ncols`): crea una figura con `nrows*ncols` areas
  - devuelve la figura y todos los ejes



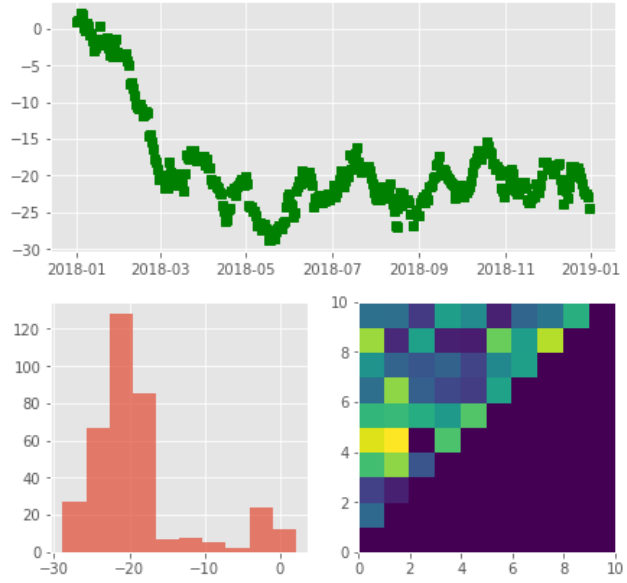

```
fig, ejes = plt.subplots(
    2, 1, sharex=True,
    figsize=(8, 10),
    facecolor='#fabada'
)
ejes[0].plot(t, y, 'g.')
ejes[1].plot(t, 1/y, '.-')
```

`plt.subplot()`

¡Ojo!: *`plt.subplot()` != `plt.subplots()`*

- (`nrows`, `ncols`, `index`): crea un eje
- devuelve 1 solo eje (al que se haga referencia)

```
fig = plt.figure(figsize=(10,

ax1 = plt.subplot(2, 1, 1)
ax1.plot(t, y, 'gs')

ax2 = plt.subplot(2, 2, 3)
ax2.hist(y, alpha=.7)

ax3 = plt.subplot(2, 2, 4)
ax3.pcolormesh(
  np.tril(np.random.uniform(size=(10, 10)),
          -1)
)
```



Guardar una figura a fichero

```
from sklearn.datasets import load_iris

iris = load_iris()
plt.style.use('ggplot')
fig, ax = plt.subplots(figsize=(7, 6))
formatter = plt.FuncFormatter(lambda i, *args: iris.target_

plt.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target
plt.colorbar(ticks=[0, 1, 2], format=formatter)

figure.savefig('iris.pdf')
```

Tipos de fichero soportados, según backend:

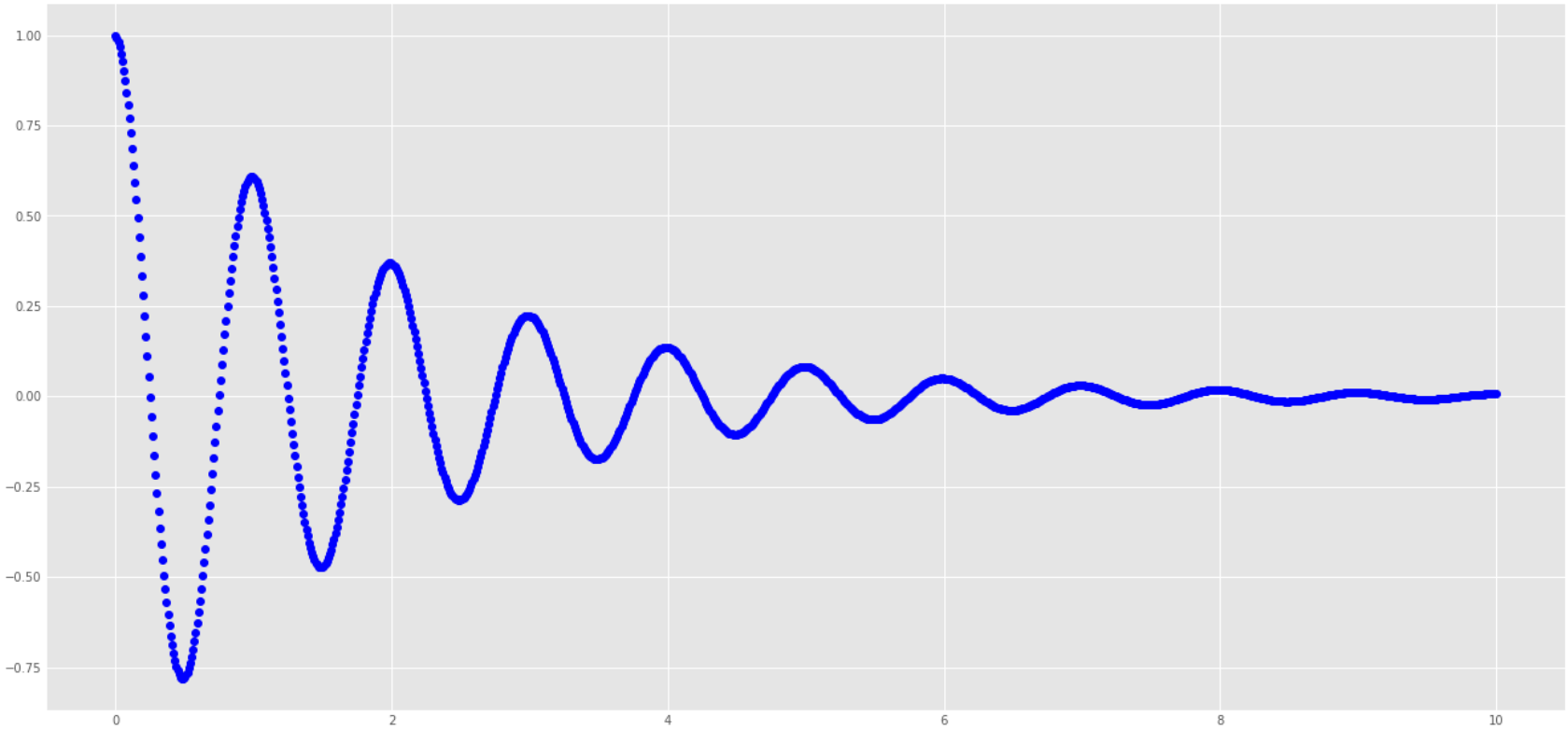
```
print(fig.canvas.get_supported_filetypes())
```

Tipos de gráficos más importantes

Lineplot (`.plot()`)

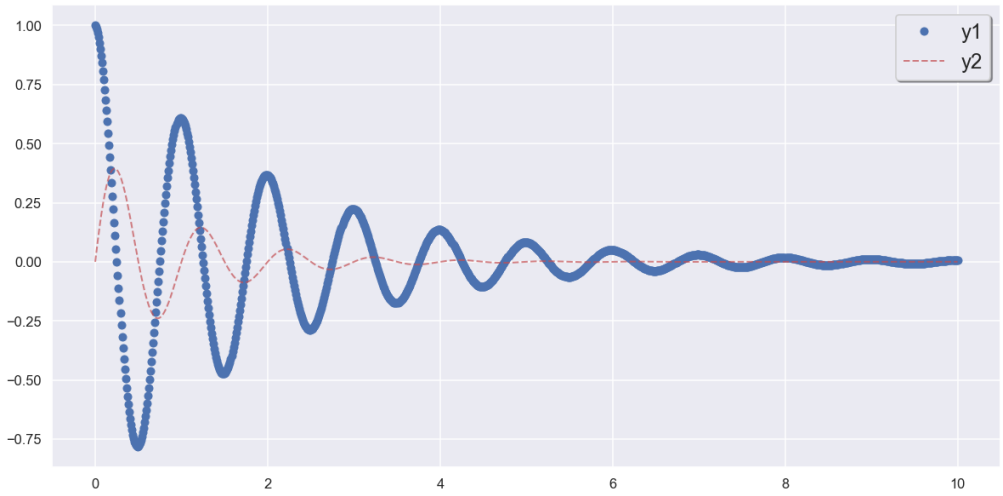
Tipo de gráfico por defecto ## ~~~python (fix, ax) = plt.subplots(1, 1) # (n_rows, n_cols)

x = np.linspace(0, 10, 1000) y = np.exp(-x/2) * np.cos(2*np.pi*x) ax.plot(x, y, 'bo') ~~~



Dos gráficos al mismo tiempo:

```
(fig, ax) = plt.subplots(1, 1)  # (n_rows, n_cols)

x = np.linspace(0, 10, 1000)
y1 = np.exp(-x / 2) * np.cos(2 * np.pi * x)
y2 = 0.5 * np.exp(-x) * np.sin(2 * np.pi * x)
ax.plot(x, y1, 'bo',
        x, y2, 'r--')
```



Dos gráficos al mismo tiempo:

```
(fig, ax) = plt.subplots(1, 1)  # (n_rows, n_cols)

x = np.linspace(0, 10, 1000)
y1 = np.exp(-x / 2) * np.cos(2 * np.pi * x)
y2 = 0.5 * np.exp(-x) * np.sin(2 * np.pi * x)
ax.plot(x, y1, 'bo')
ax.plot(x, y2, 'r--')
```

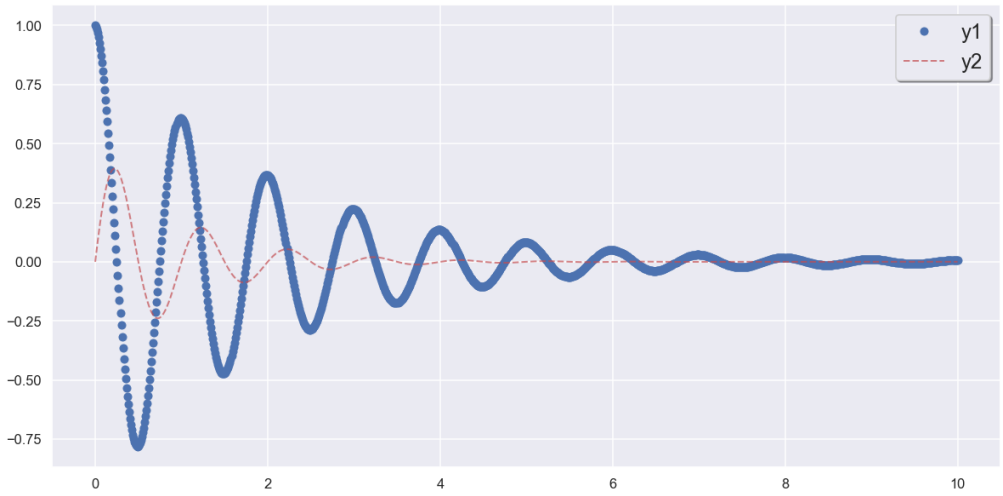


Gráfico de dispersión

```
N = 75
np.random.seed(45987230)



fig, (ax1, ax2) = plt.subplots(2, 1
                                    figs:
a = np.random.randint(low=1, high=1:
b = a + np.random.randint(1, 5, size
x = np.linspace(0, 1, N)
t = np.random.gamma(5, size=N)
colors = np.random.rand(N)

ax1.scatter(x=a, y=b, marker='o', c=
            edgecolor='b')
ax1.set_title('$a$ vs $b$')
ax2.scatter(
    x, t,
    s=np.random.randint(10,800, N),  # tamaño
    marker='v',  # tipo de marcador
    c=colors,  # colores
```

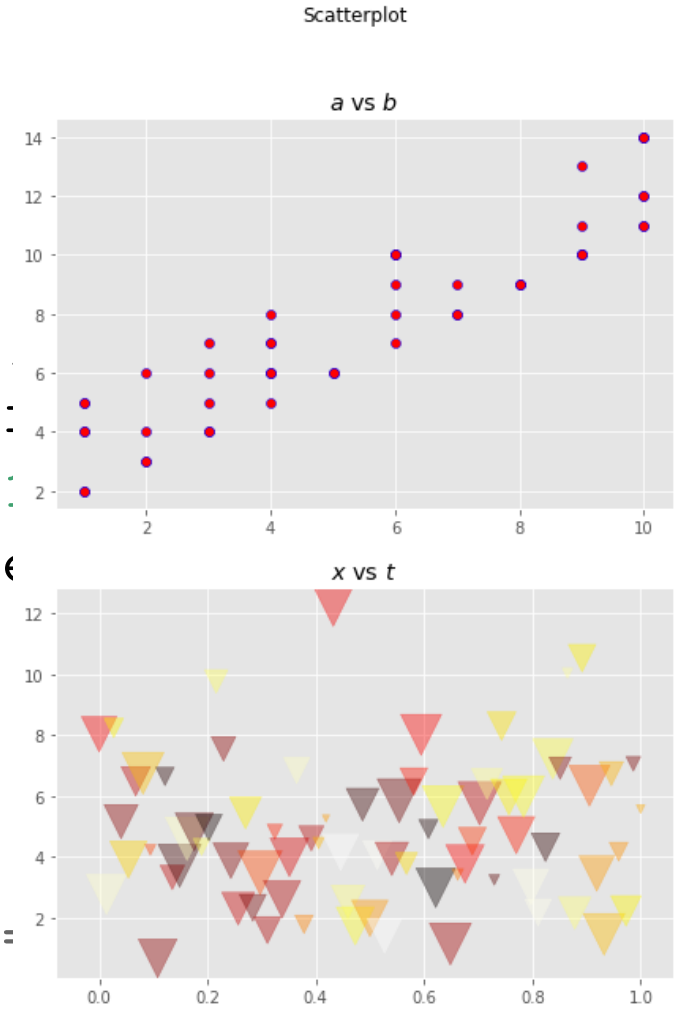
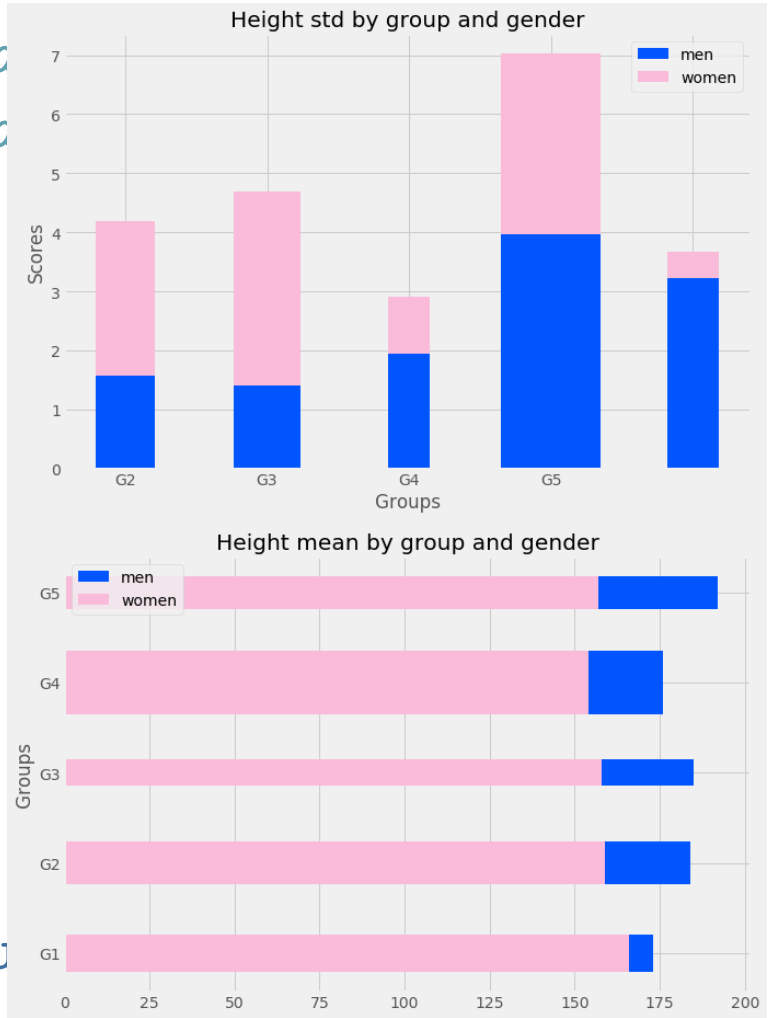


Gráfico de barras

```
ax1.bar(ind,        # eje de abcisc
        men_std,    # eje de ordena
        width,      # anchura
        color='#0055ff')
ax1.bar(ind,
        women_std,
        width,
        color='#fabada',
        bottom=men_std)

ax1.set_ylabel('Scores')
ax1.set_xlabel('Groups')
ax1.set_title('Height std by grou
ax1.legend(['men', 'women'], loc='best')

# Barras horizontales

ax2.barh(ind,           # eje de ordenadas
         men_means,     # eje de abcisas
         width,         # anchura
```



Histograma

```
fig, ax = plt.subplots(2, figsize=(10,5))
bins = 20
x1 = np.random.gamma(10, size=1000)
x2 = np.random.randn(1000)


ax1, ax2 = ax.flatten()

(ax1_values, _, _) = ax1.hist(x1, bins=bins, facecolor='br
(_, ax2_bins, _) = ax2.hist(x2, alpha=.7, cumulative=True,
                            log=True, orientation='horizont
```

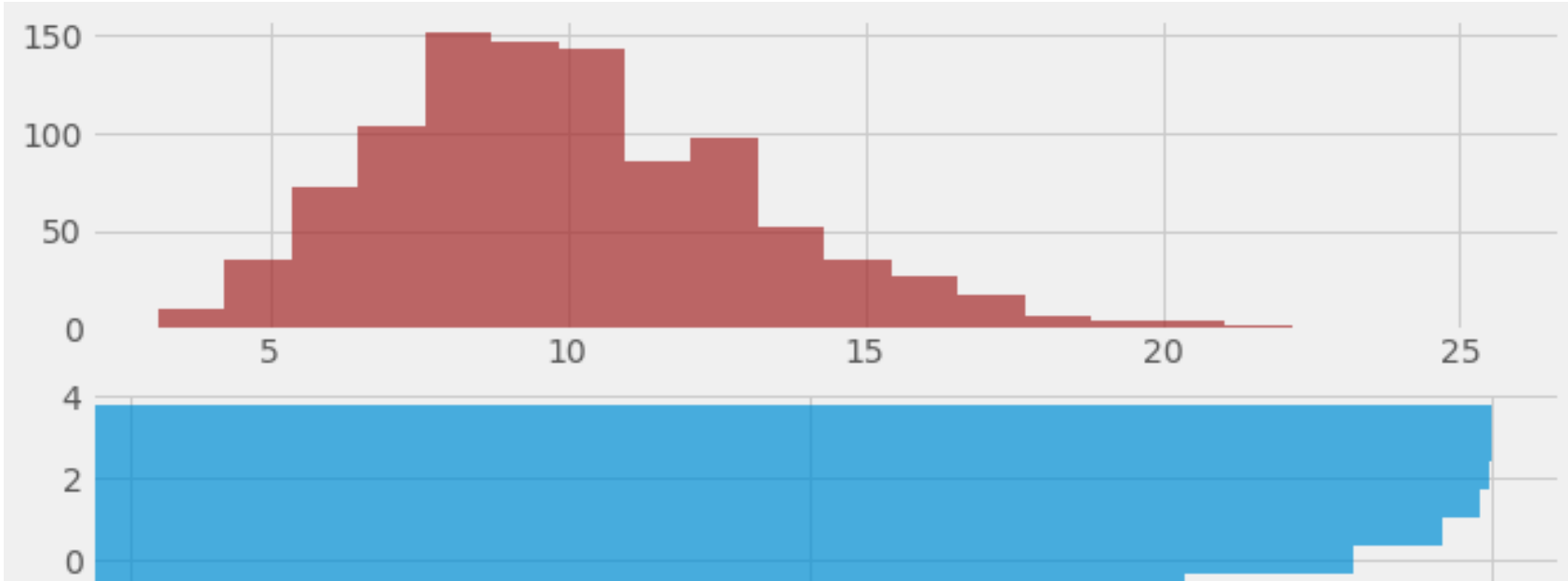
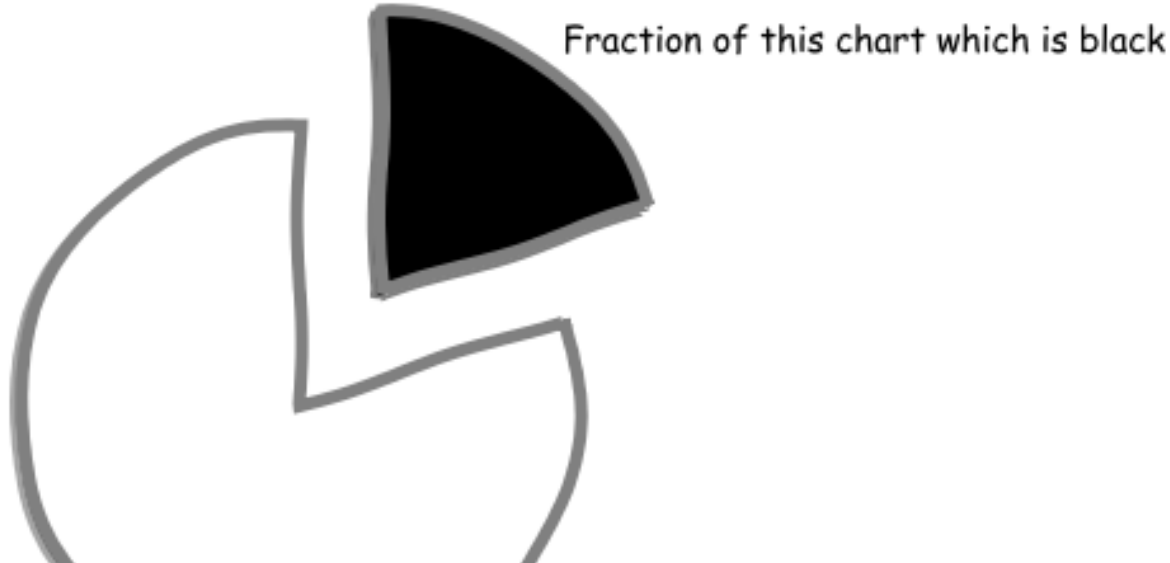


Diagrama circular

```
colors = ['white', 'black']
labels = [f'Fraction of this image which is {color}'
          for color in colors]

ax.pie(
    [80, 20], explode=(0, .5), labels=labels,
    colors=colors, shadow=True, startangle=90
)
# ejes iguales = asegurarnos de que se muestre
# como un círculo
ax.axis('equal');
```



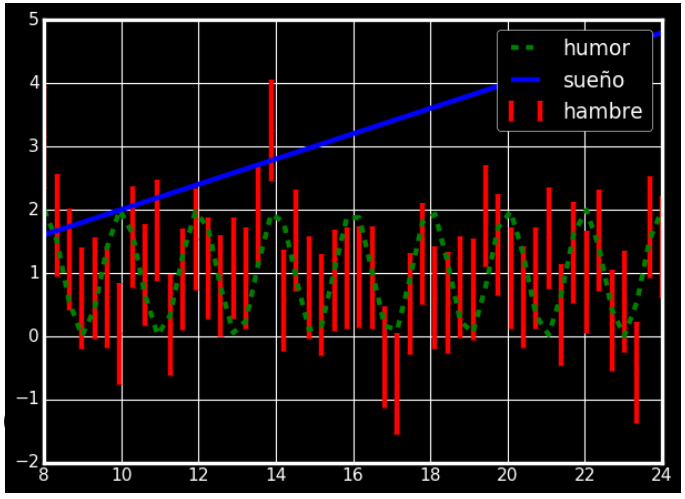
Elementos y parámetros

Color/estilo de línea

El color de línea para un gráfico individual se puede controlar mediante una cadena de texto (p.e. `'r--'`):

```
colors = {'b': 'blue', 'g': 'green', 'r': 'red', 'c': 'cyan
          'y': 'yellow', 'k': 'black', 'w': 'white'}

lineStyles = {'-': '_draw_solid', '--': '_draw_dashed', '-
              ':': '_draw_dotted', 'None': '_draw_nothing'
              ' ': '_draw_nothing', '': '_draw_nothing'}

markers = {
    '.': 'point', ',': 'pixel', 'o': 'circle', 'v': 'triang
    '^': 'triangle_up', '<': 'triangle_left', '>': 'triangl
    '1': 'tri_down', '2': 'tri_up', '3': 'tri_left', '4':
    '8': 'octagon', 's': 'square', 'p': 'pentagon', '*': 's
    'h': 'hexagon1', 'H': 'hexagon2', '+': 'plus', 'x': 'x
    'd': 'thin_diamond', '|': 'vline', '_': 'hline', 'P':
    'X': 'x_filled', 0: 'tickleft', 1: 'tickright', 2: 'tic
    4: 'caretleft', 5: 'caretright', 6: 'caretup', 7: 'care
```

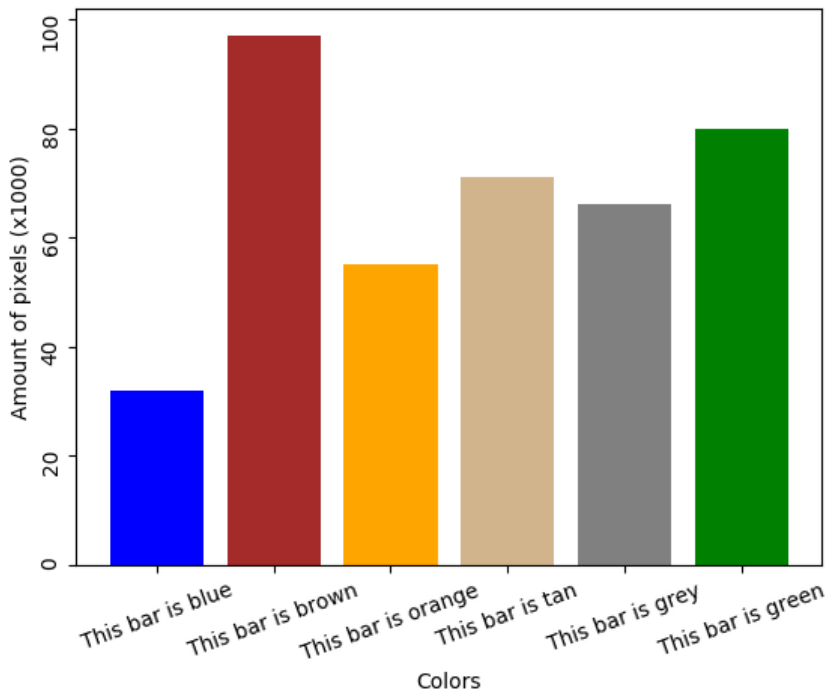
```
N = 50
np.random.seed(4873)

x = np.linspace(0, 10, N)
k = 0.8
y = k + np.sin(x) * np.random.randn(
```



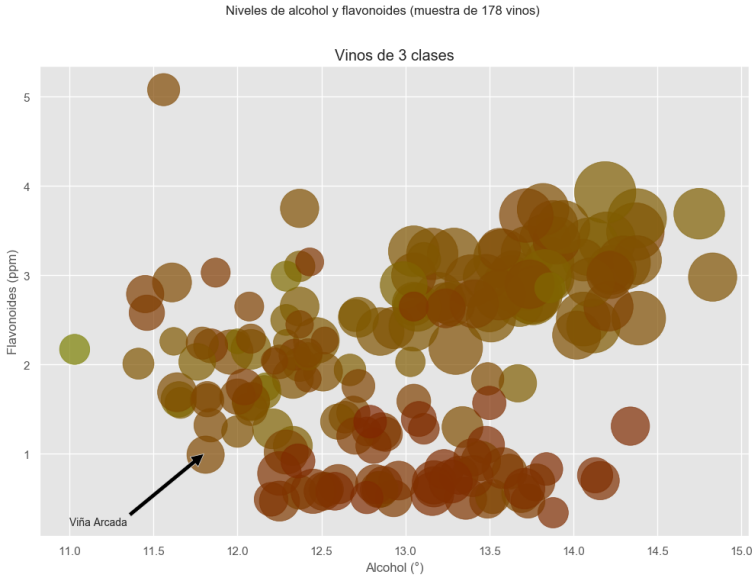

```
(fig, ax) = plt.subplots(1)
ax.errorbar(x, y, yerr=k, fmt='.r');
ax.plot(x, 1 + np.cos(np.pi*x), '--g')
ax.plot(x, x/5, 'b')
```

xticks/yticks

```
plt.yticks(rotation='vertical')
plt.xticks(rotation=20)   # 20 grados en sentido antihorari
```



Títulos, etiquetas y anotaciones



- Título de la figura
- Título de cada subplot (`Axes`)
- Título de cada eje (`Axis`)
- Anotaciones

```
fig.suptitle('Niveles de alcohol y f
                '(muestra de 178 vinos)
ax.set_title('Vinos de 3 clases')
ax.set_xlabel('Alcohol (°)')
ax.set_ylabel('Flavonoides (ppm)')
ax.annotate(
    'Viña Arcada', xy=(11.8, 1),
     xytext=(11, .5),
    arrowprops=dict(facecolor='black
)
```

Leyenda

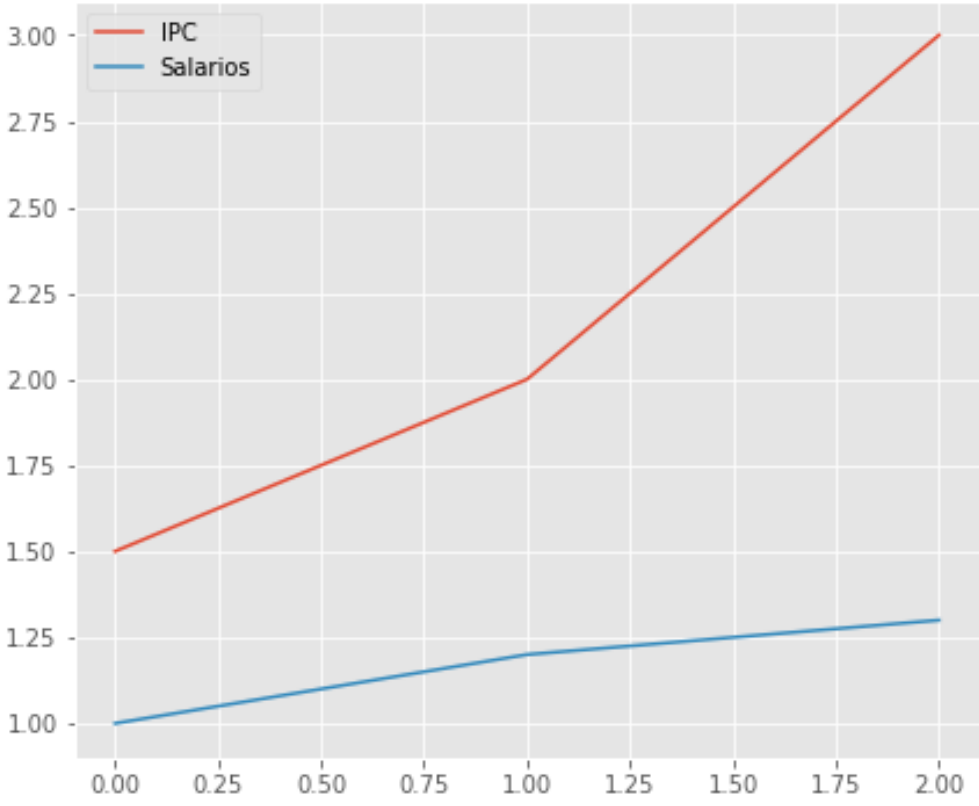
- Se genera automáticamente según datos inferidos de las etiquetas

```
(line1, ) = ax.plot([1.5,
                     label
(line2, ) = ax.plot([1, 1
                     label
ax.legend(loc='upper left
```

`loc`: define dónde emplazar la leyenda

- ***best***, *upper right*, *upper left*, *lower left*, *lower right*, *right*, *center left*, *center right*, *lower center*, *upper center*, *center*

:::



Estilos

- Aplican a todos los gráficos generados
- Estilo por defecto en `matplotlib.rcParams`
  - pueden modificarse dinámicamente
- Podemos cambiar a estilos preconfigurados:

```
print(plt.style.available)
plt.style.use('ggplot')
```

- y/o modificar parámetros individualmente

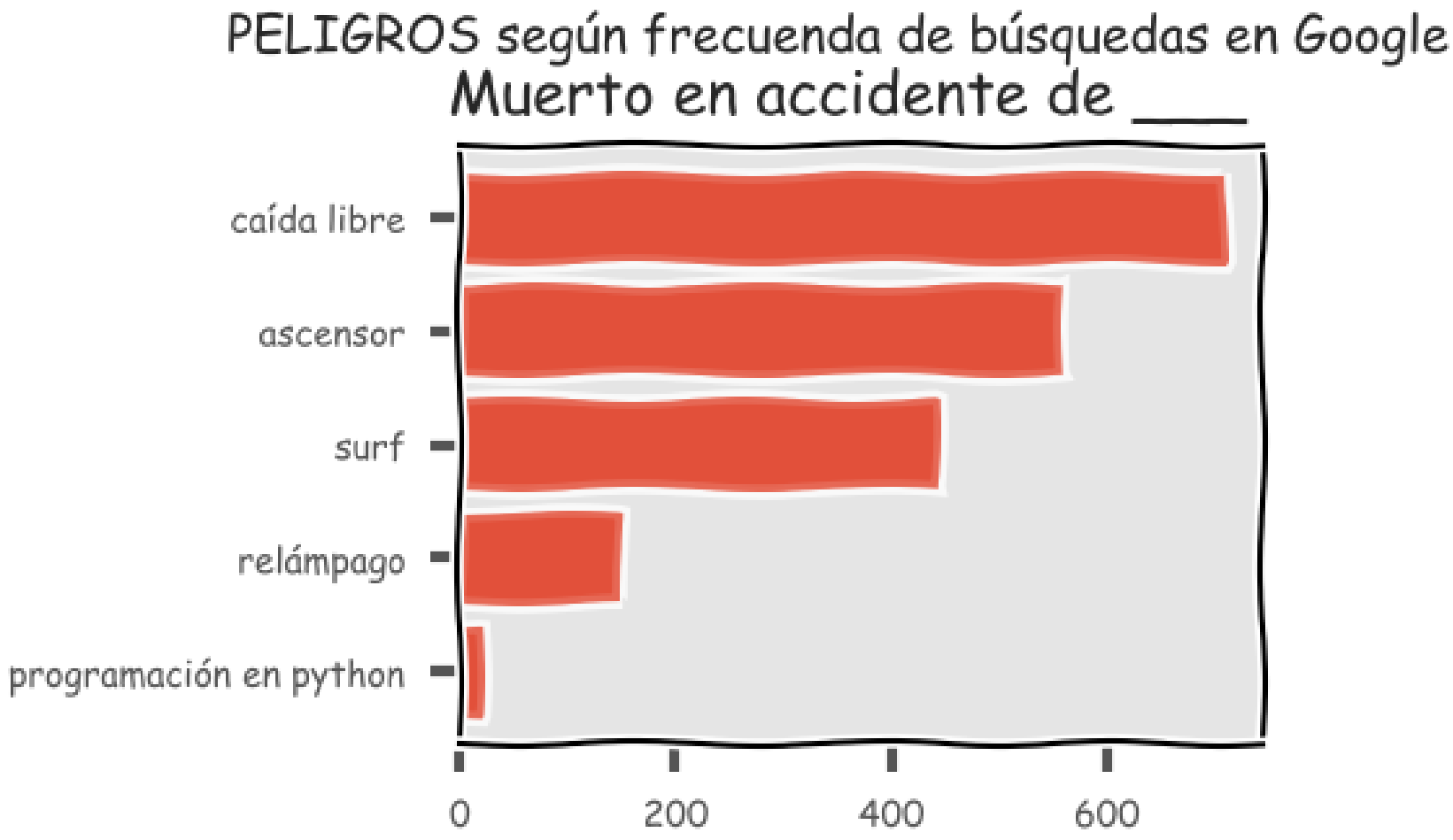
```
plt.rcParams["figure.figsize"] = (20.0, 15.0)
plt.rcParams['font.family'] = ['monospace']
```

- Los cambios aplican a todos los gráficos
  - salvo cambio temporal de estilos:

```
with plt.style.context(('dark_background')):
    plt.plot(np.sin(np.linspace(0, 2 * np.pi)), 'r-o')
```

Otros estilos

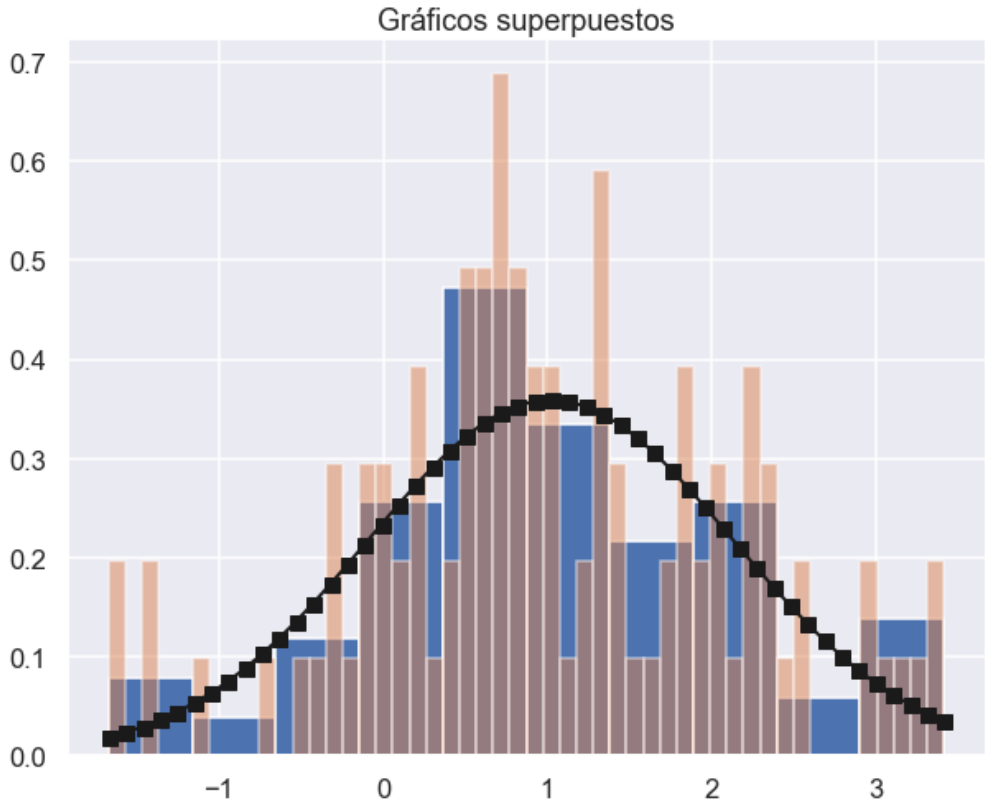
```
with plt.xkcd():
    ...
```



Otros estilos

seaborn

```
import seaborn as sns

sns.set()
sns.set_context("talk")
```



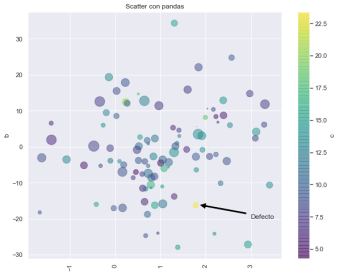
:::

Integración con pandas

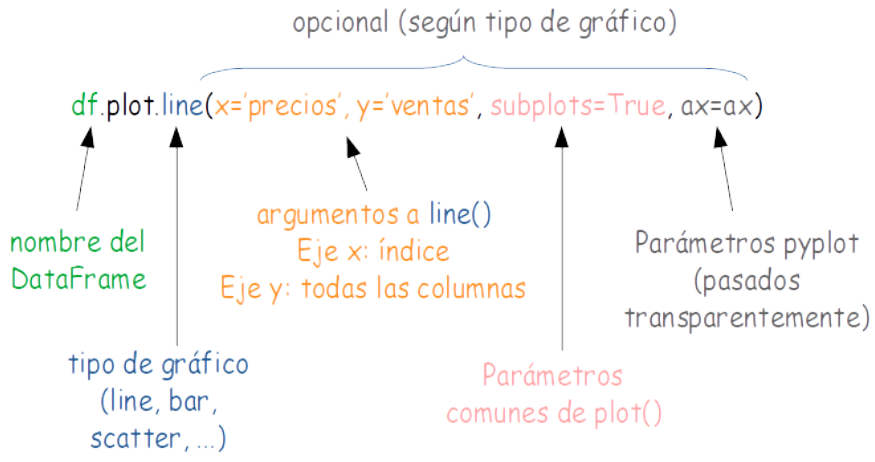
pyplot está (parcialmente) integrado en pandas

- Podemos dibujar directamente desde un dataframe de pandas
- se devuelve un objeto "*Axes*" sobre el que poder trabajar
  - no es necesario crear con antelación la figura y los ejes
  - ... aunque suele ser lo recomendable

```
# Crea la figura y los ejes
fig, ax = plt.subplots()

# siempre se devuelve referencia al eje
ax2 = df.plot.scatter(
    'a', 'b', c='c', s=df['d'],
    colormap='viridis', alpha=.5,
    title='Scatter con pandas', rot='vertical',
    ax = ax;
)

ax.annotate(
    'Defecto', xy=(1.9, -17),
    xytext=(2, -20),
```



Sintaxis



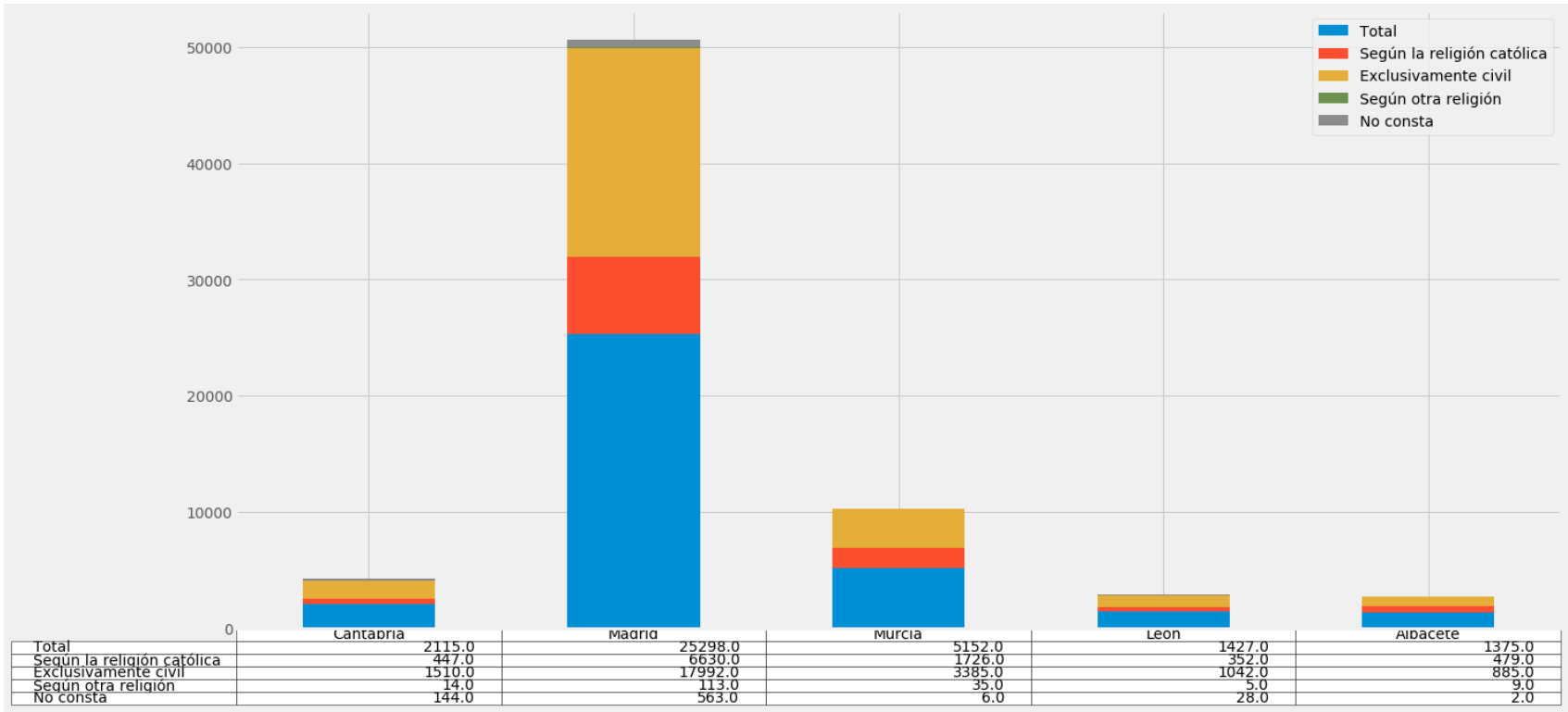
De forma alternativa:

```
df.plot(kind='line', x='precios', y='ventas', subplots=True, ax=ax)
```

Visualizar datos usando la interfaz de pandas es conveniente y mucho más sencillo:

```
provincias = ['Cantabria', 'Madrid', 'Murcia', 'León', 'Alb
index = np.arange(len(provincias)) + 0.3

y_offset = np.zeros(len(marriages.Total.columns))
cell_text = []

for row in marriages.Total.columns:
    _data = marriages.Total.loc[provincias, row]
    plt.bar(index, _data, bottom=y_offset, width=0.5)
    y_offset = y_offset + _data
    cell_text.append(["%1.1f" % (x / 1000.0) for x in y_off
cell_text.reverse()
tabla = plt.table(cellText=cell_text,
                  rowLabels=marriages.Total.columns,
                  colLabels=provincias,
                  loc='bottom')
plt.legend(marriages.Total.columns)
```

```
provincias = ['Cantabria', 'Madrid', 'Murcia', 'León', 'Alb
df = ax.Total.loc[provincias]
ax = df.plot.bar(
    stacked=True,
    table=True,
    title='Total Matrimonios en 2017'
)
```



	Cantabria	Madrid	Murcia	Leon	Albacete
Total	2115.0	25298.0	5152.0	1427.0	1375.0
Según la religión católica	447.0	6630.0	1726.0	352.0	479.0
Exclusivamente civil	1510.0	17992.0	3385.0	1042.0	885.0
Según otra religión	14.0	113.0	35.0	5.0	9.0
No consta	144.0	563.0	6.0	28.0	2.0

Generalmente las opciones más usadas en cada tipo de gráfico (título, ejes y posición, color, tamaño de línea, ...) son directamente accesibles desde `pandas.DataFrame.plot()`.

Para el resto, usar `matplotlib` refiriéndonos al objeto a modificar:

```
ax = df.plot(kind='scatter', ...)  # df.plot.scatter(...)
ax.set_yticks(rotation='vertical')
```

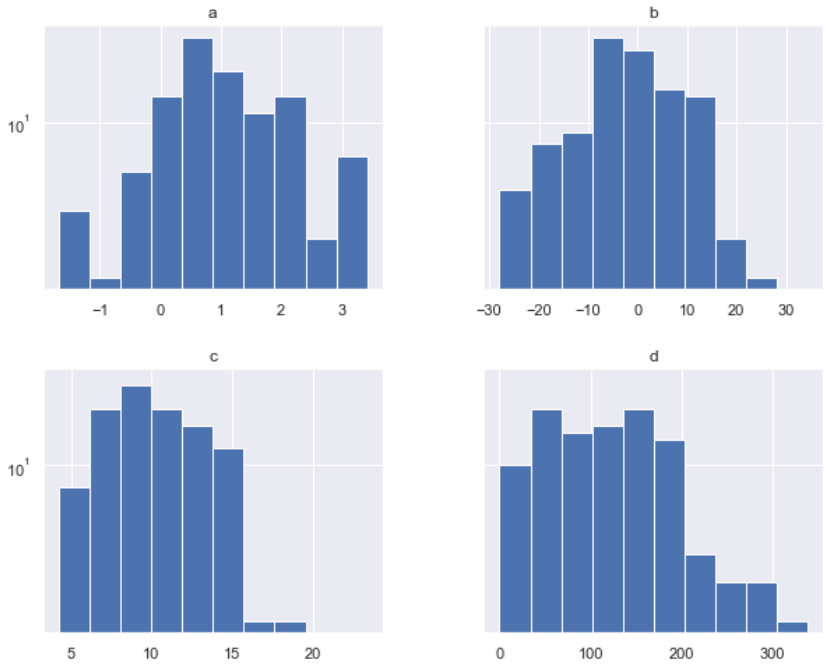
Tipos de gráficos disponibles

- Gráfico de dispersión (`scatter`)

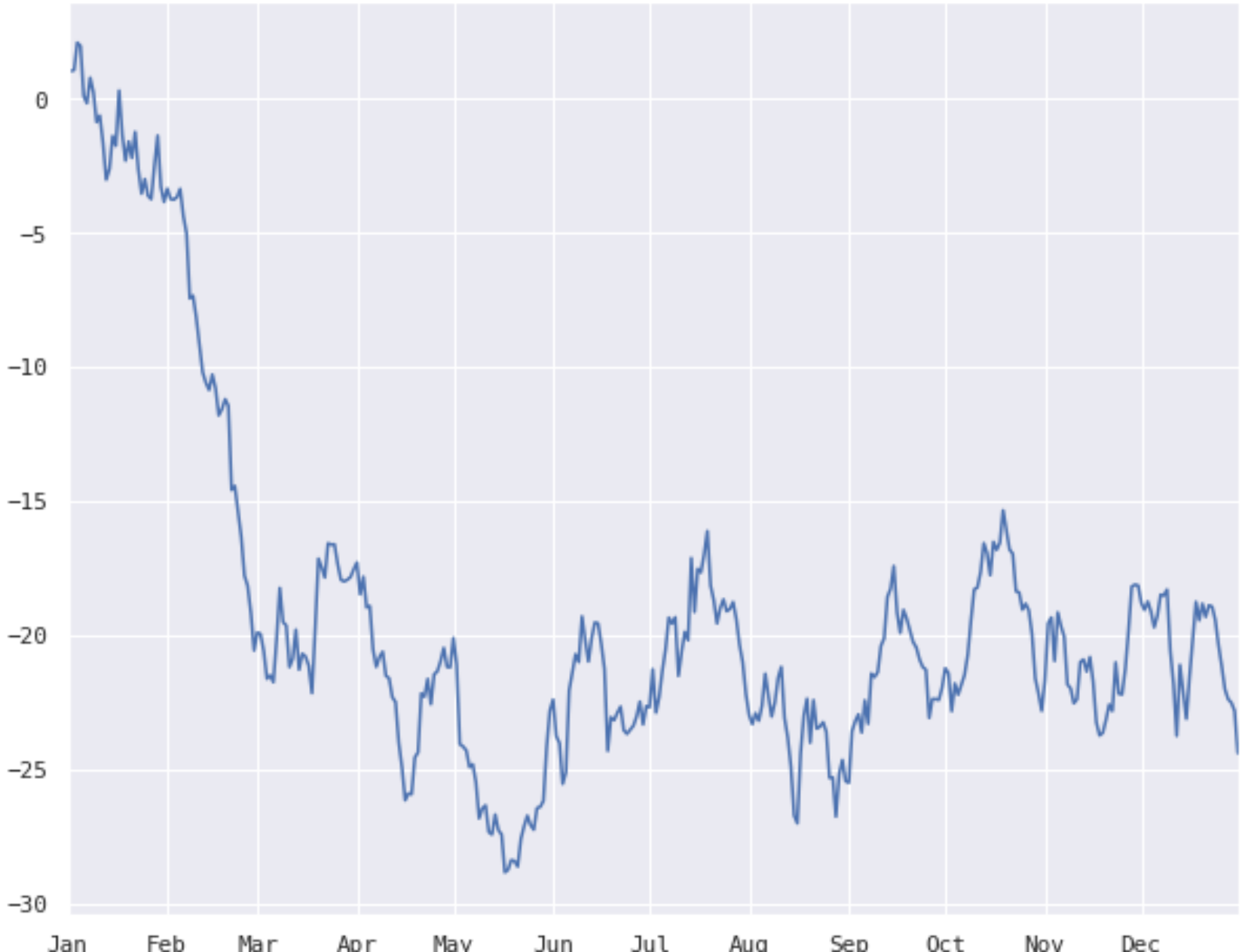
```
df.plot.scatter(
    x='a', y='b',  # nombres de las columnas del dataframe
    c='c',  # columna con datos de color
    s=df['d'],  # tamaño de los puntos
    colormap='viridis',  # paleta de color
    alpha=.5,  # transparencia
    title='Scatter con pandas',  # título
    rot='vertical',  # rotar etiquetas del eje 'x'
)
```

- Histograma

```
df.hist(sharey=True, log=True)
```

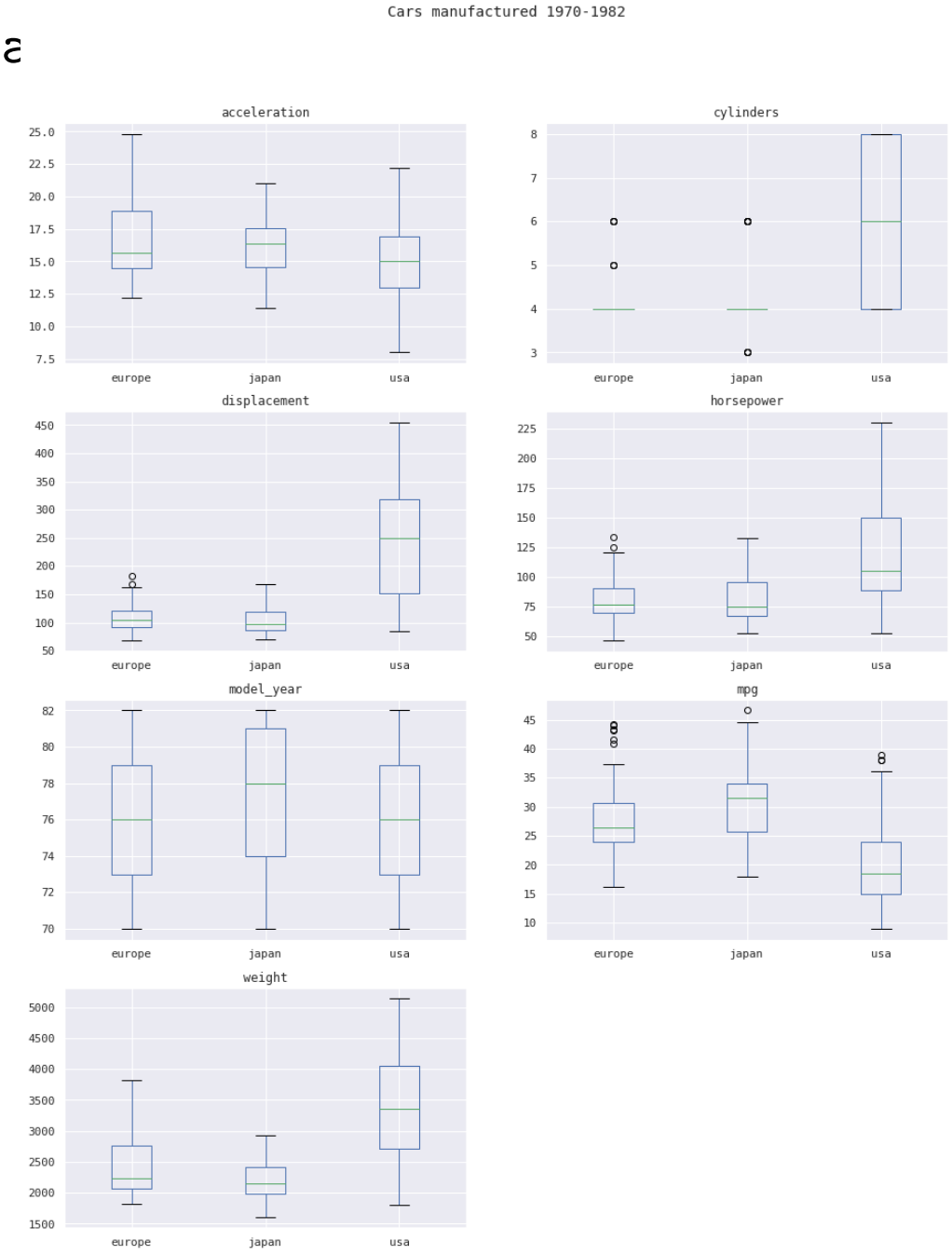


- **Series temporales**: aquellos donde el índice del `DataFrame` tiene propiedades de "índice temporal"



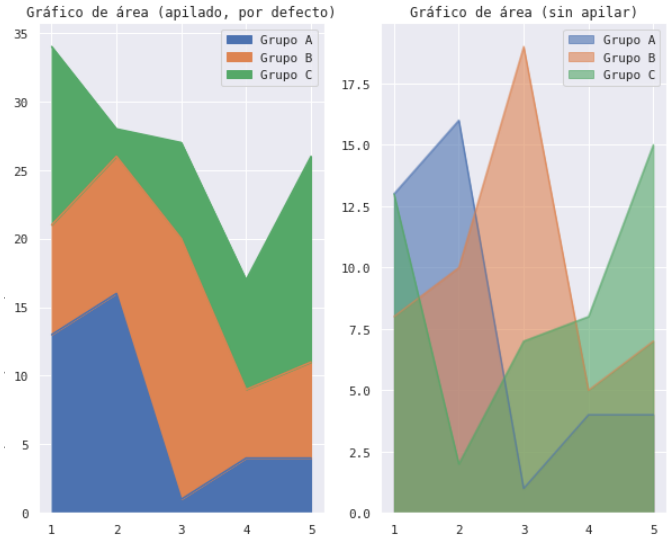
Boxplots

```
df.boxplot(by='origin', a
```



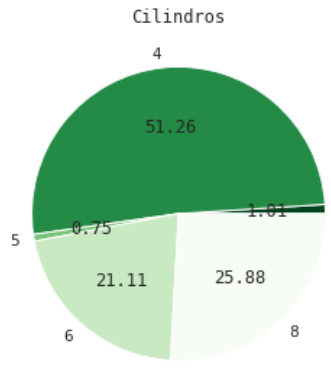
- Área

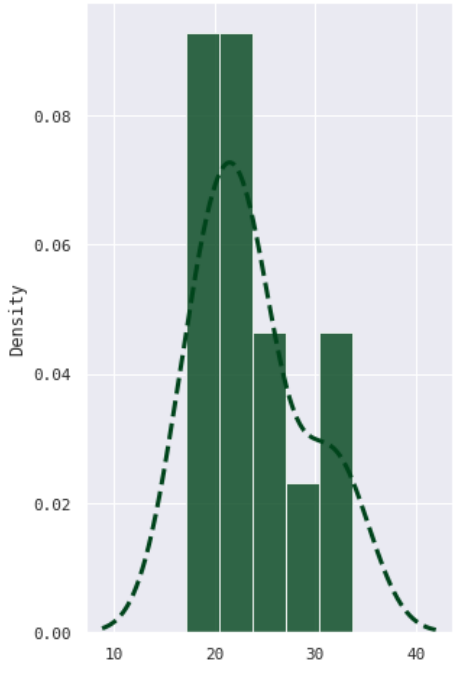
```
np.random.seed(0)
N = 5
data = pd.DataFrame(
    {'Grupo A': np.random.randint(1
     'Grupo B': np.random.randint(1
     'Grupo C': np.random.randint(1
    index=range(1, N+1)
)

ax = plt.subplot(1, 2, 1)
data.plot.area(
    stacked=False,
    alpha=.6,
    title='Gráfico de área',
    ax=ax)

ax = plt.subplot(1, 2, 2)
data.plot.area(
```



- Circular

```
ax = df1.plot.pie(cmap=cmap,
                  title='Cilin
                  autopct='%.2
```





Parámetros opcionales

`pd.DataFrame.plot()`

Es posible pasar argumentos para realizar personalizaciones rápidas.

- Existen opciones específicas para cada tipo de visualización
- También hay opciones **comunes** a todos ellos
- Además, podremos usar las primitivas de `matplotlib`, bien como argumentos adicionales o sobre los ejes

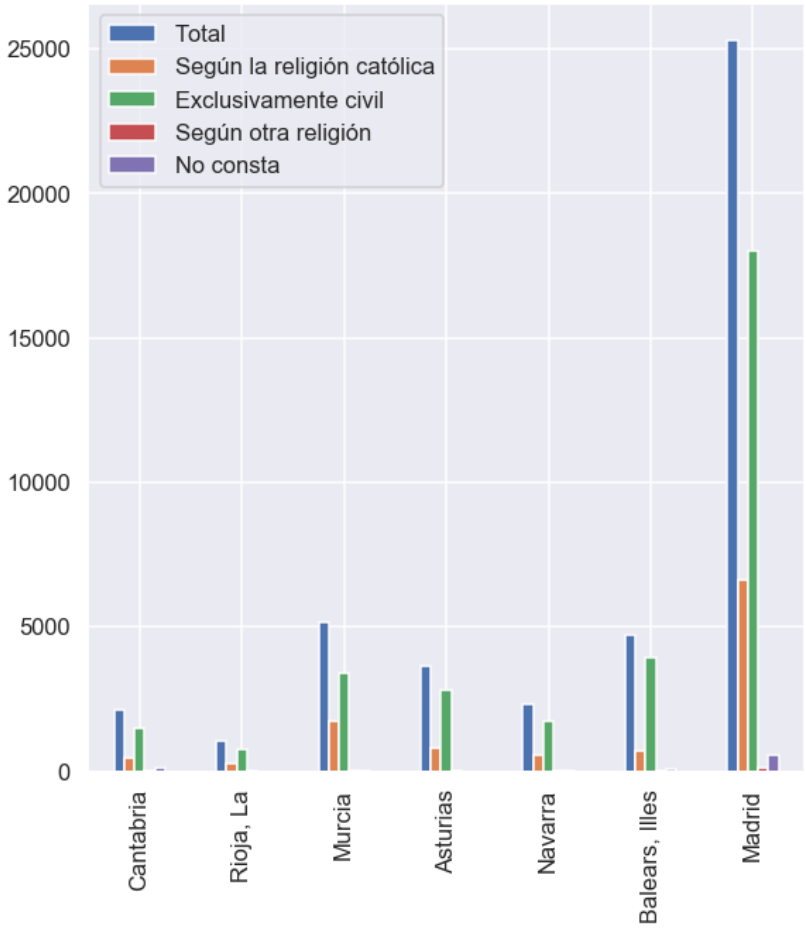
NaN

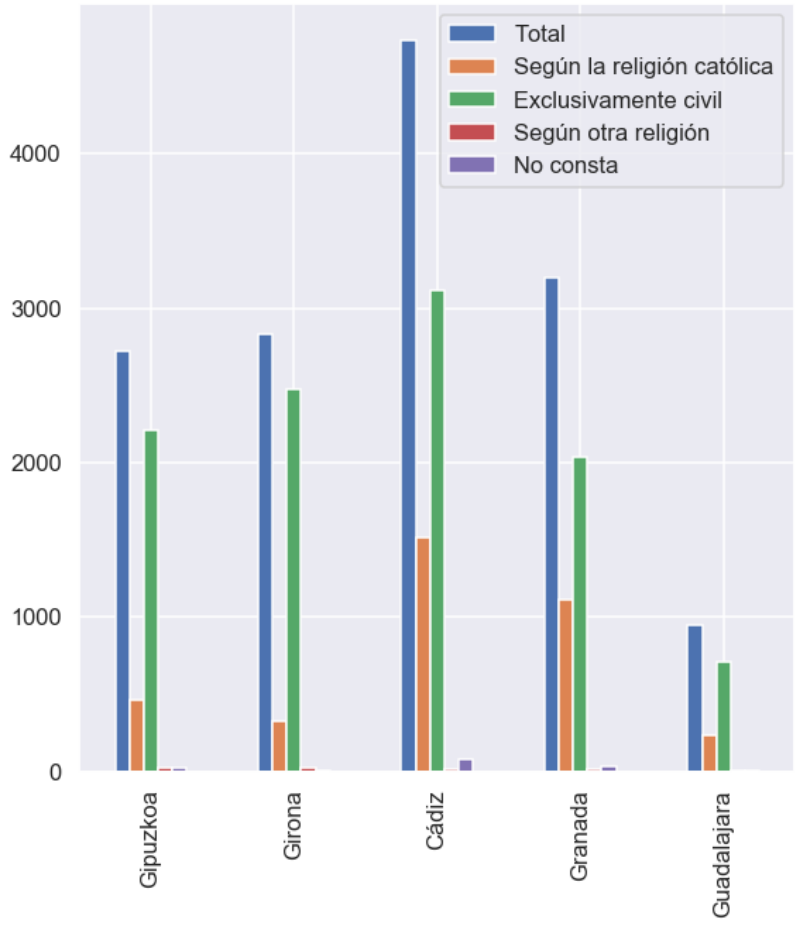
- Por defecto ignora los datos ausentes (`NaN`)
- Podemos "rellenar los huecos": `df.fillna()`
  - propagando el último valor
  - o con valores preestablecidos
- O interpolar `df.interpolate()`

Interactivo/Práctico

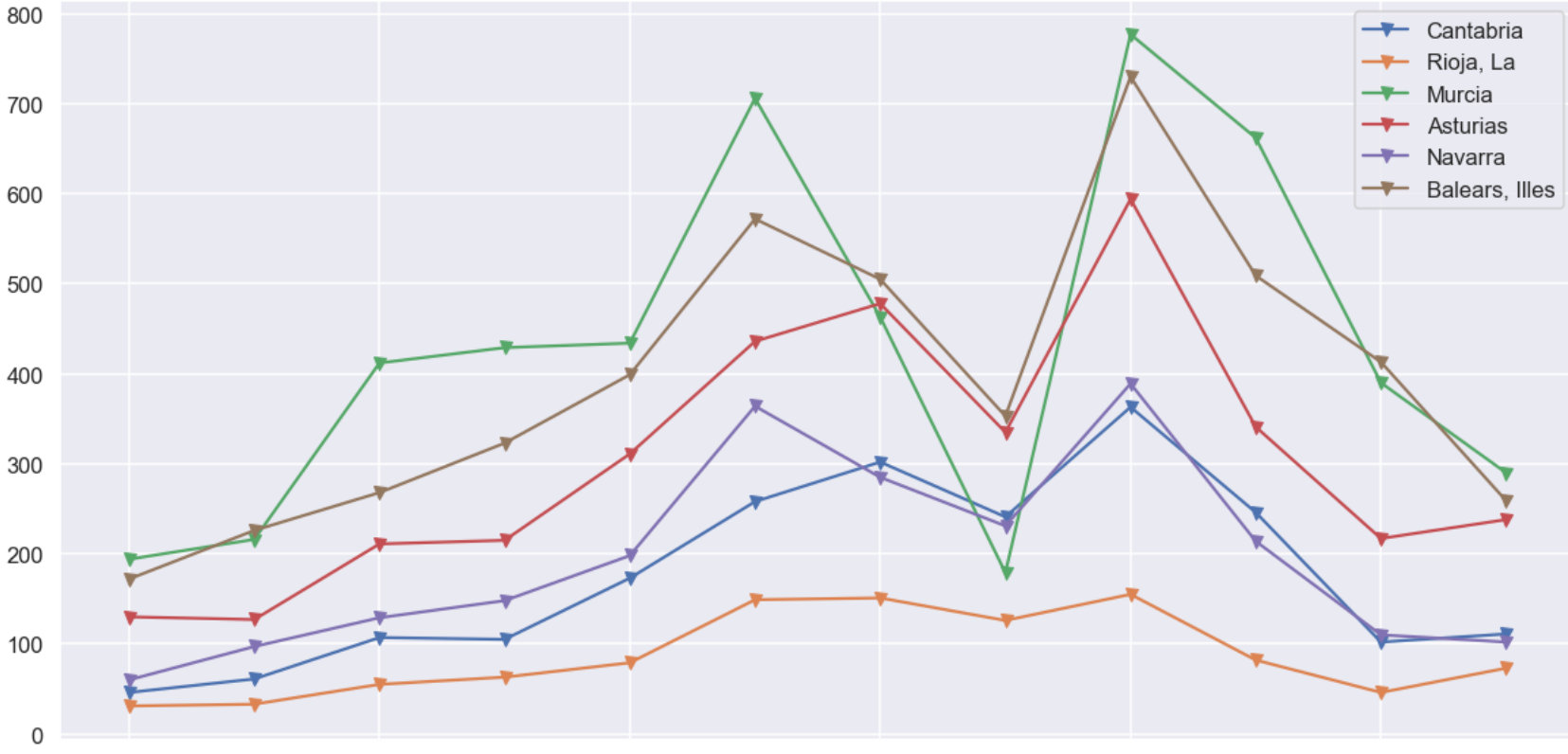
Abrir [cuaderno jupyter][nb_pract]

1. ax: ejes sobre los que dibujar

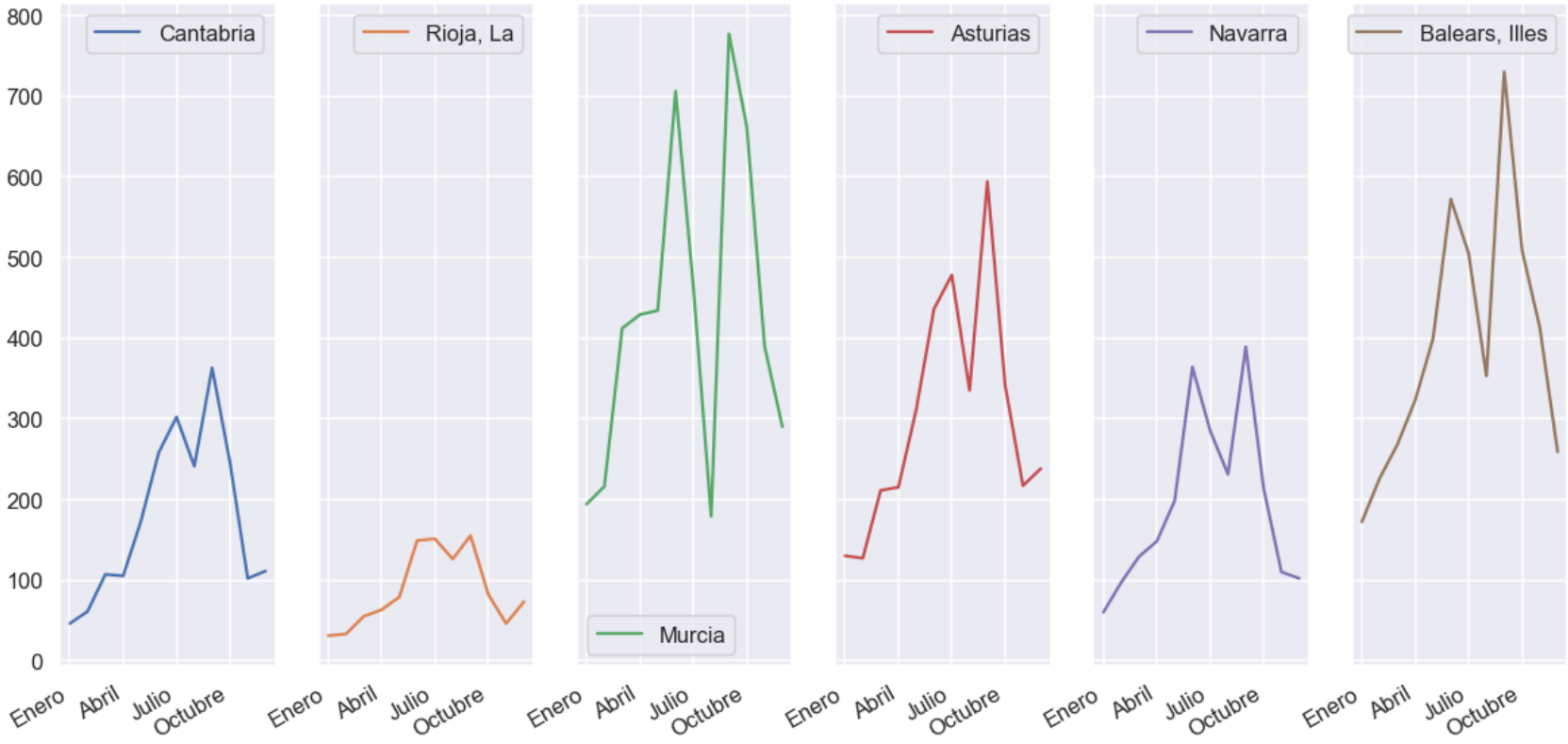




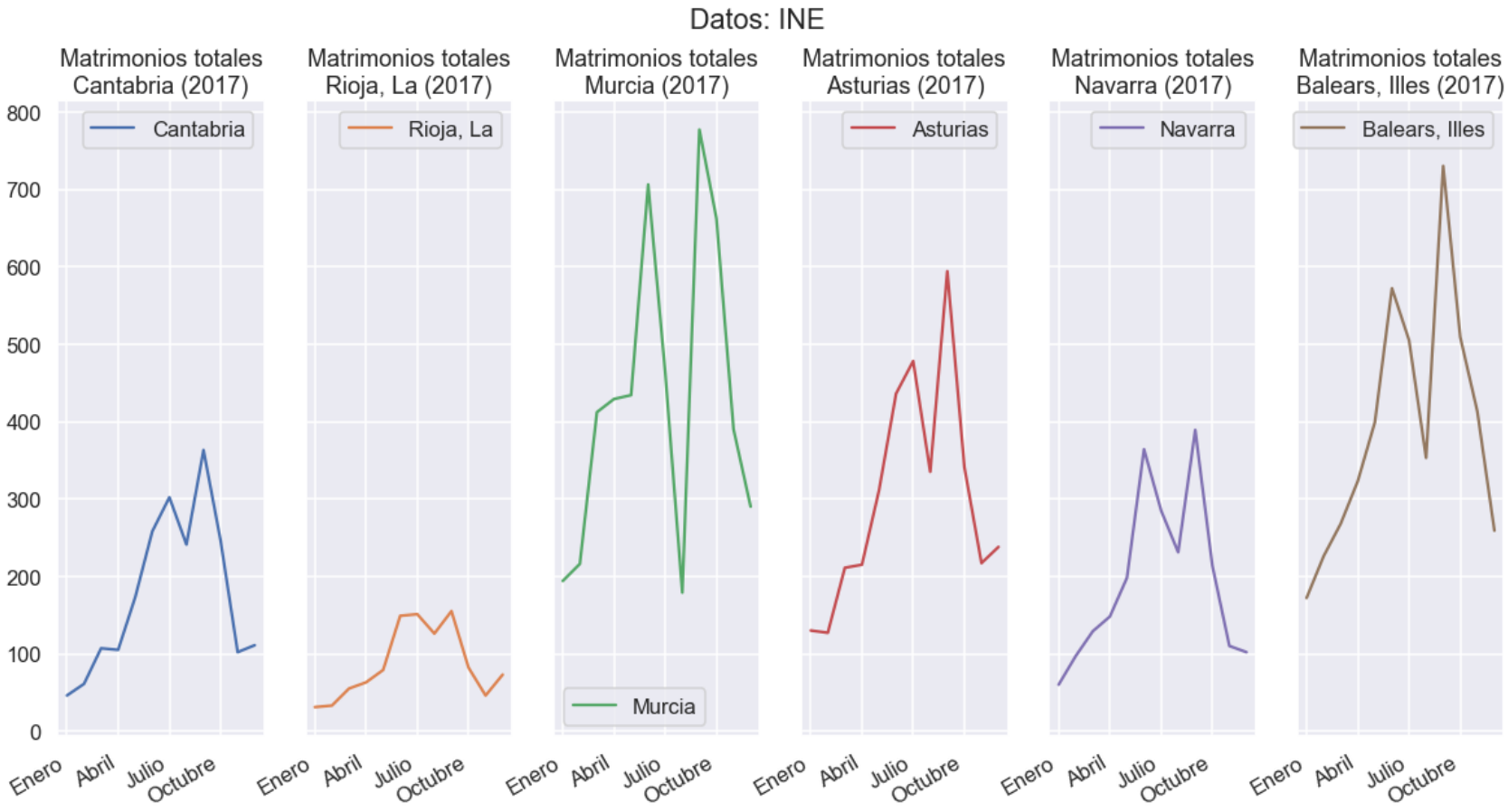
2. `subplots [True|False]`: subplot para cada columna dibujada
   - `layout`: (`n_filas`, `n_columnas`), opcional con `subplot=True`



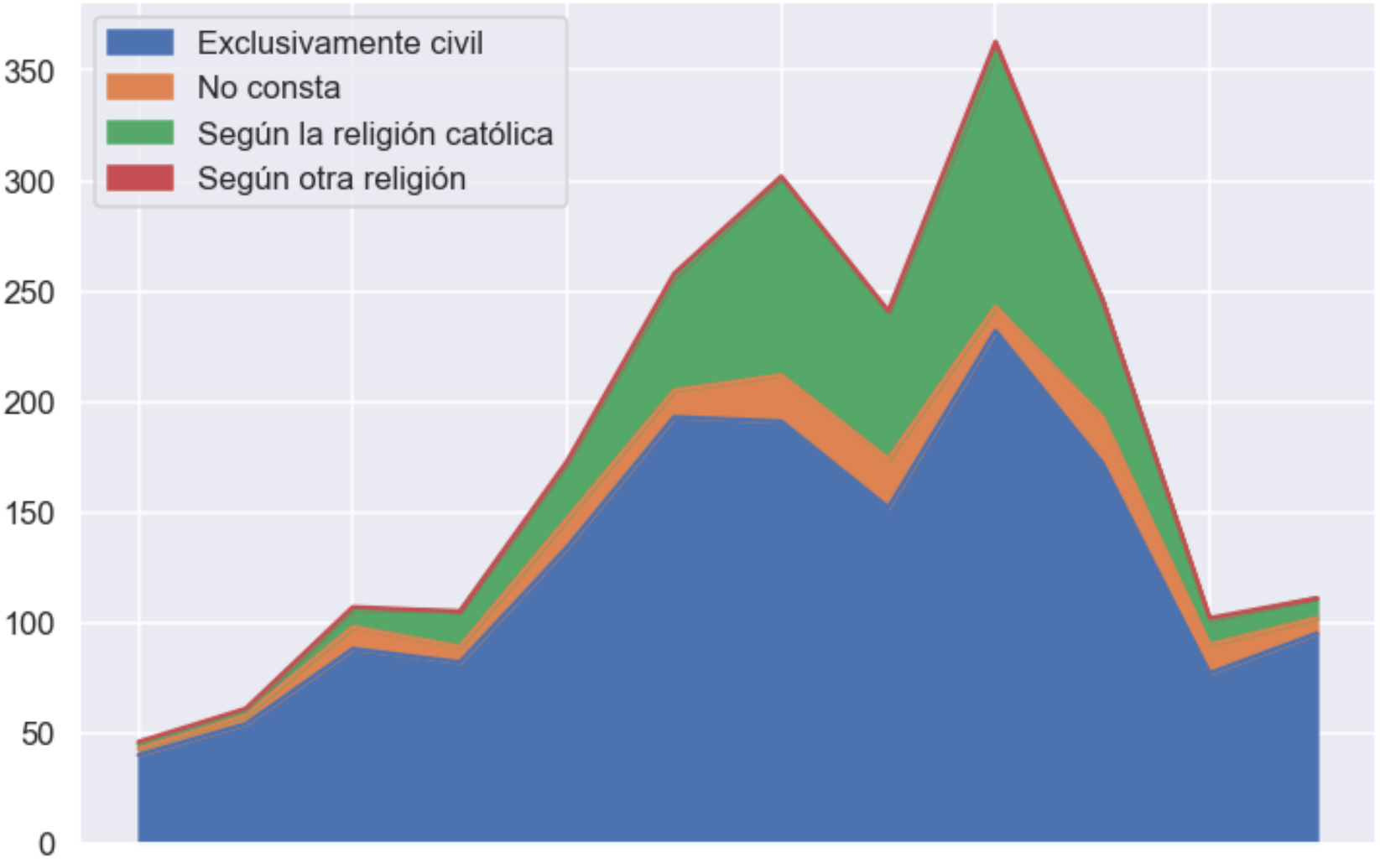
3. `sharex/sharey`: compartir ejes (subplots)



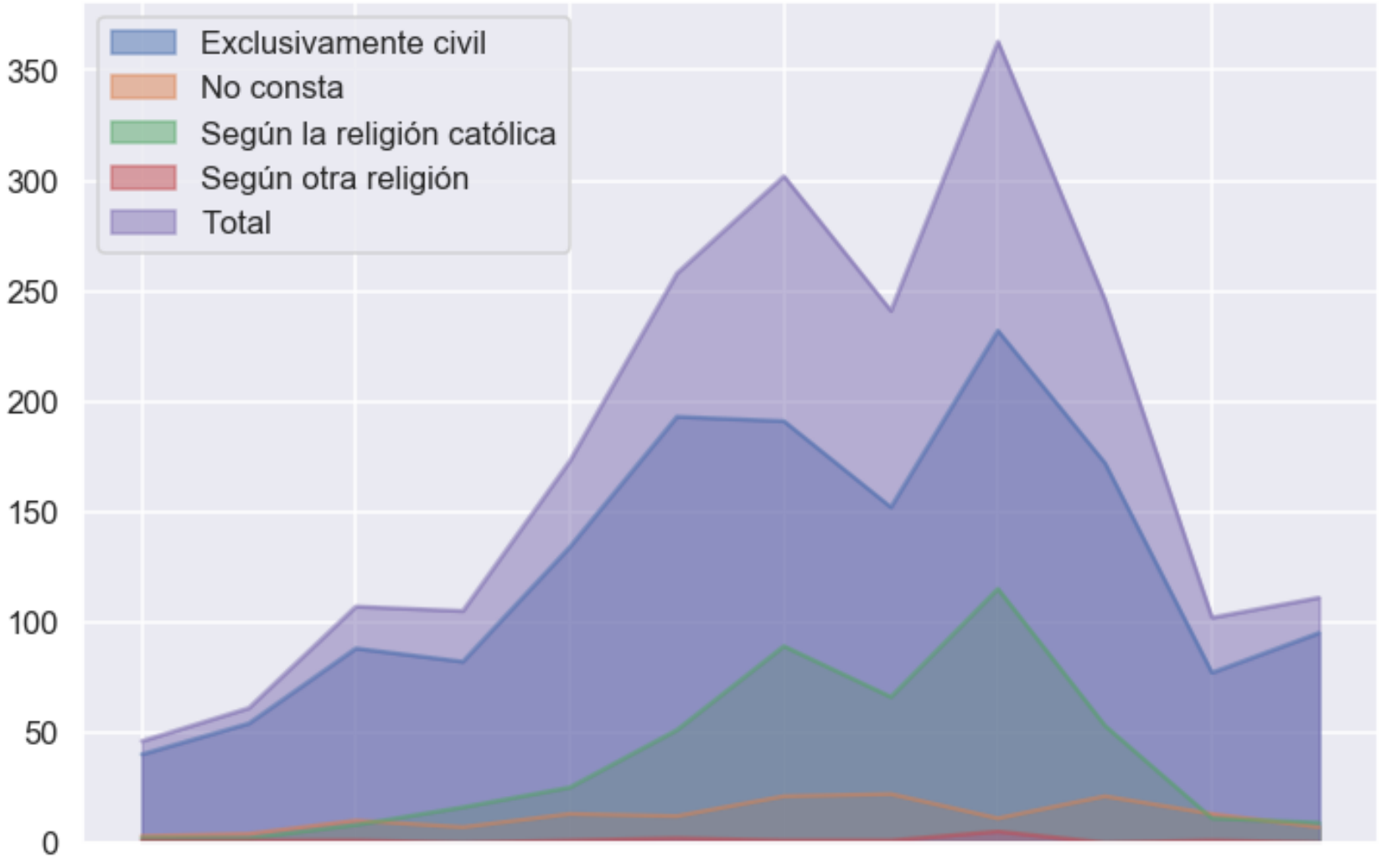
4. `title`:
   - `str`: título de la figura
   - `list(str)`: título de cada subplot



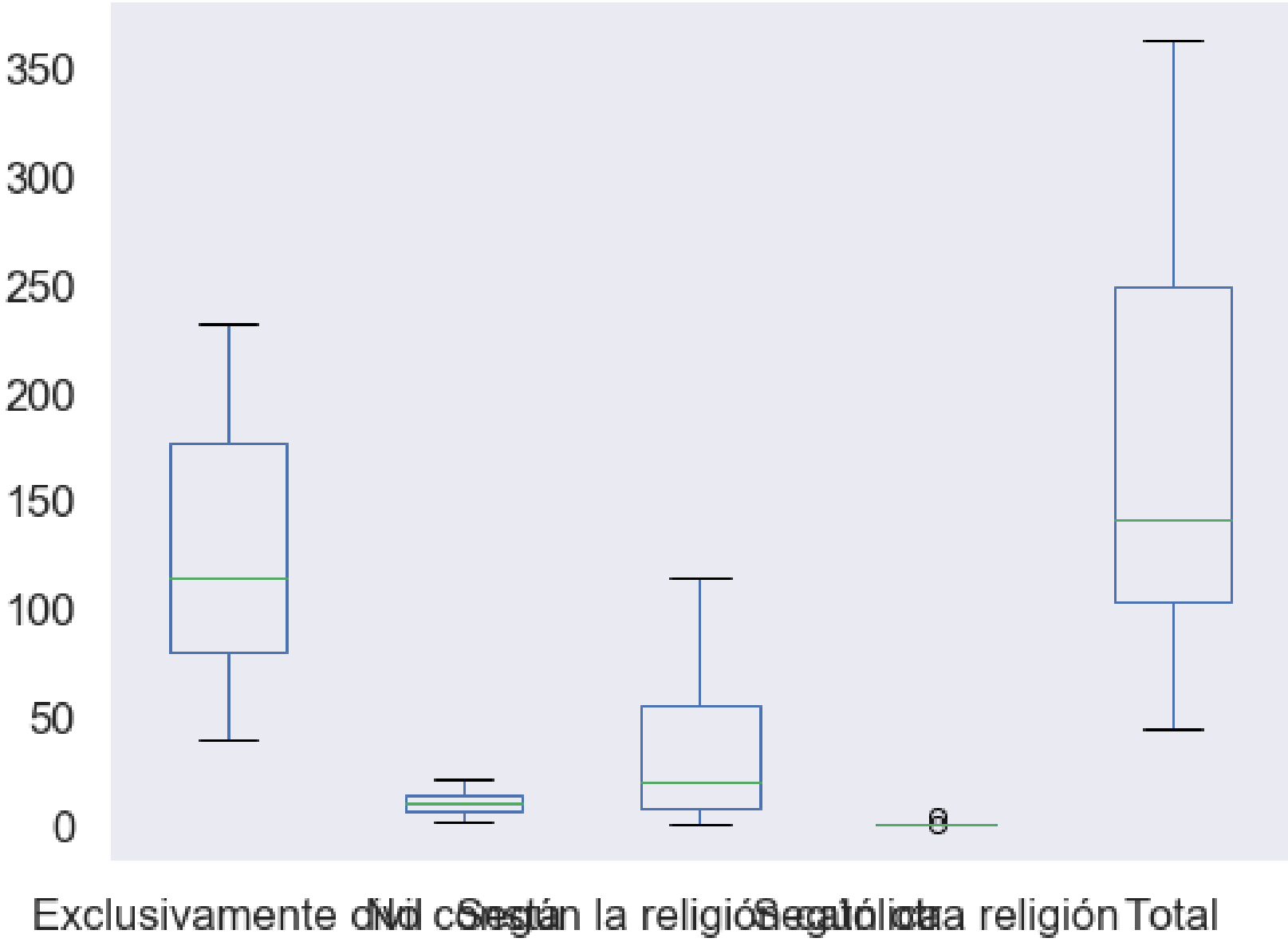
5. `figsize`: tamaño de la figura



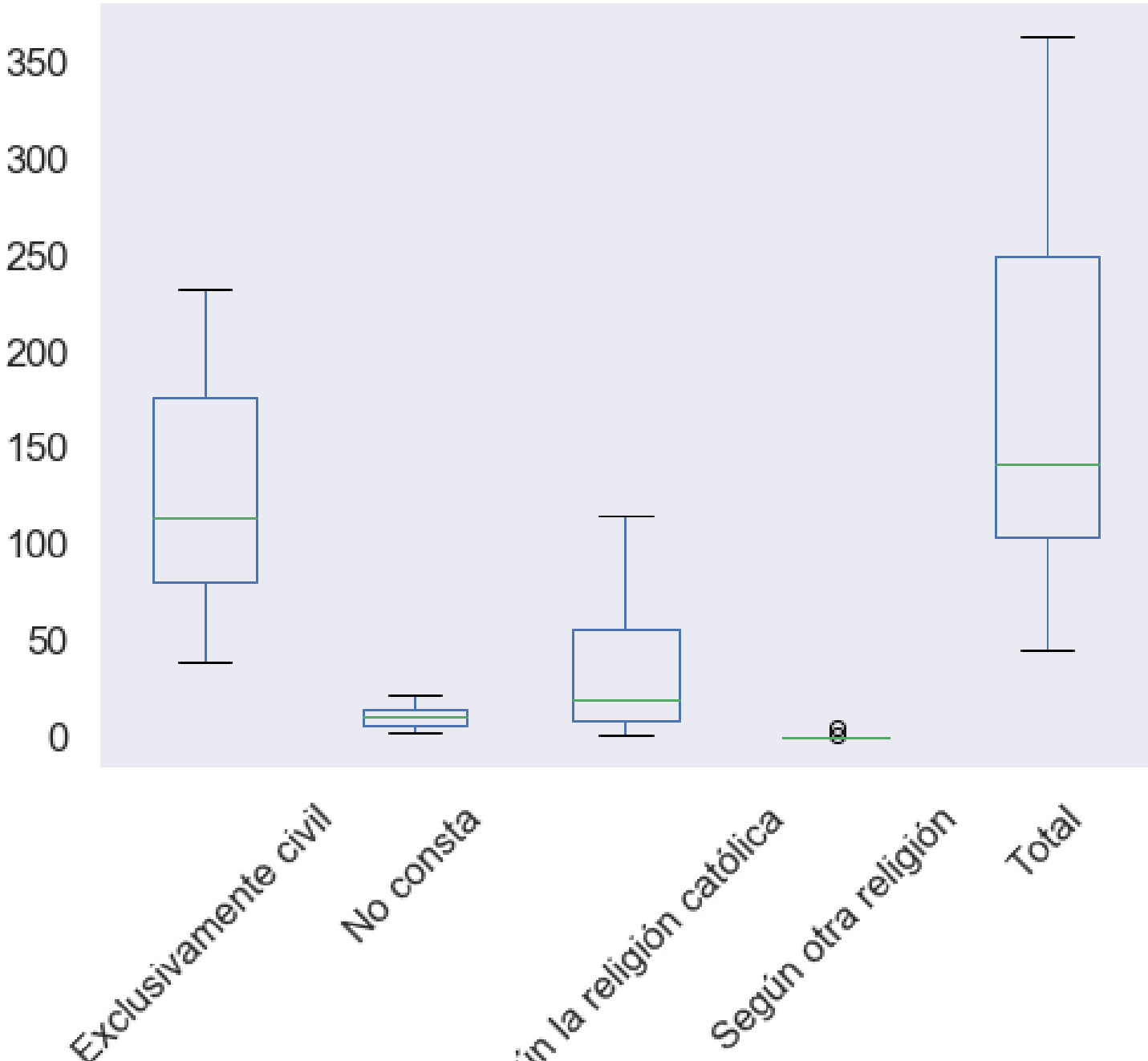
6. `stacked [True|False]`: apilar datos



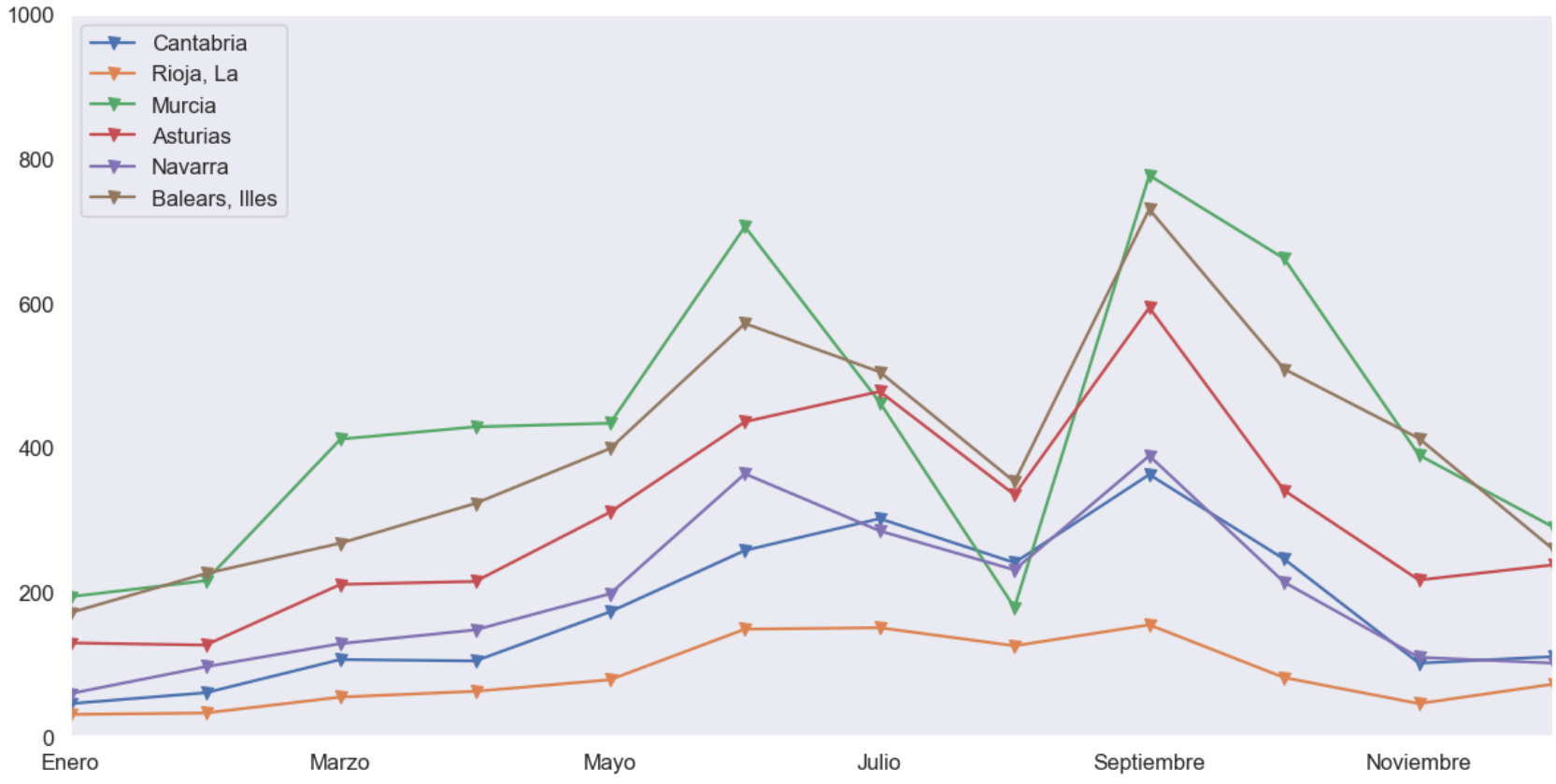
7. `grid [True|False]`: dibujar cuadrícula



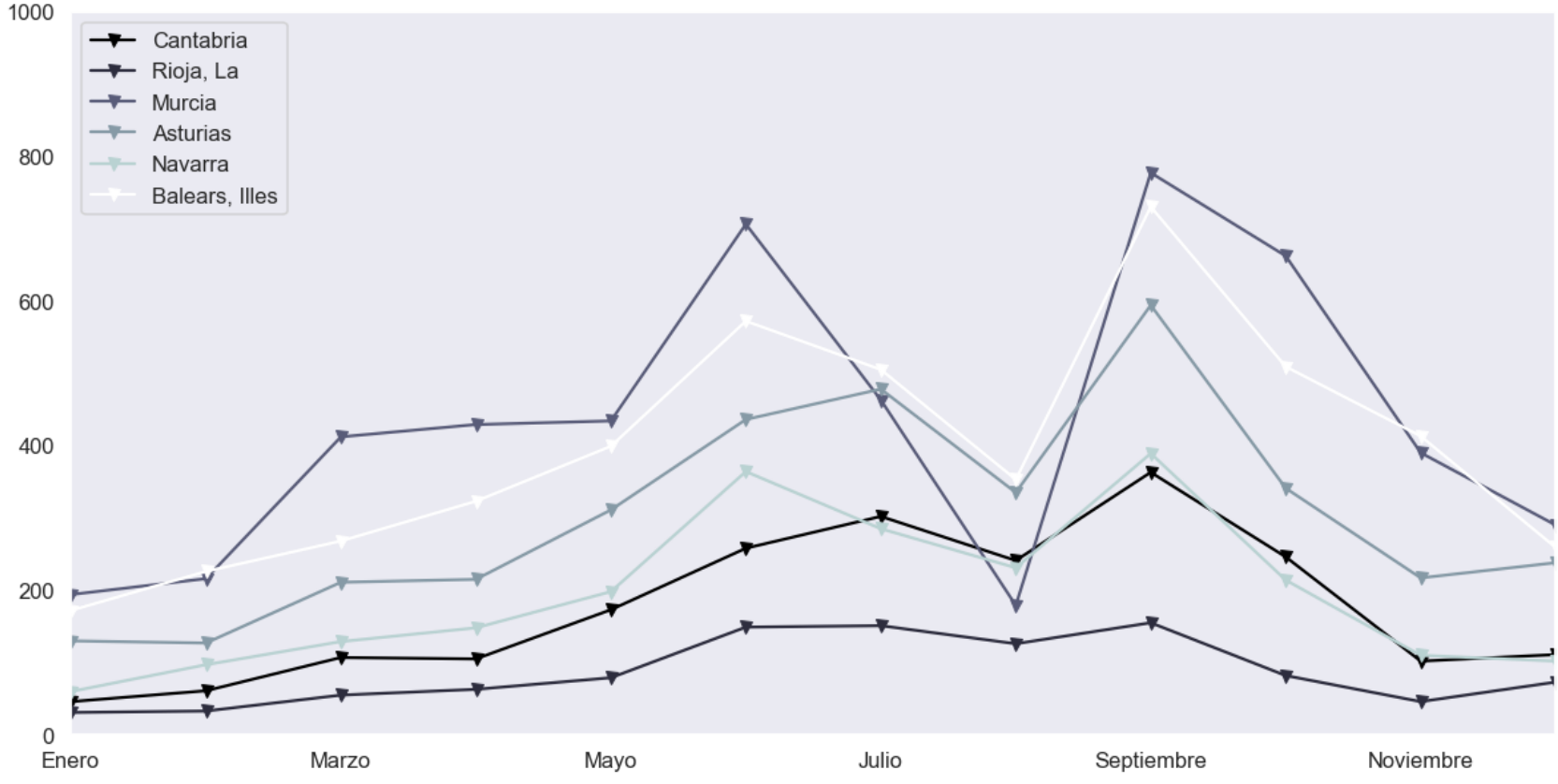
8. `rot`: `'horizontal'`, `'vertical'`, o número (en grados)



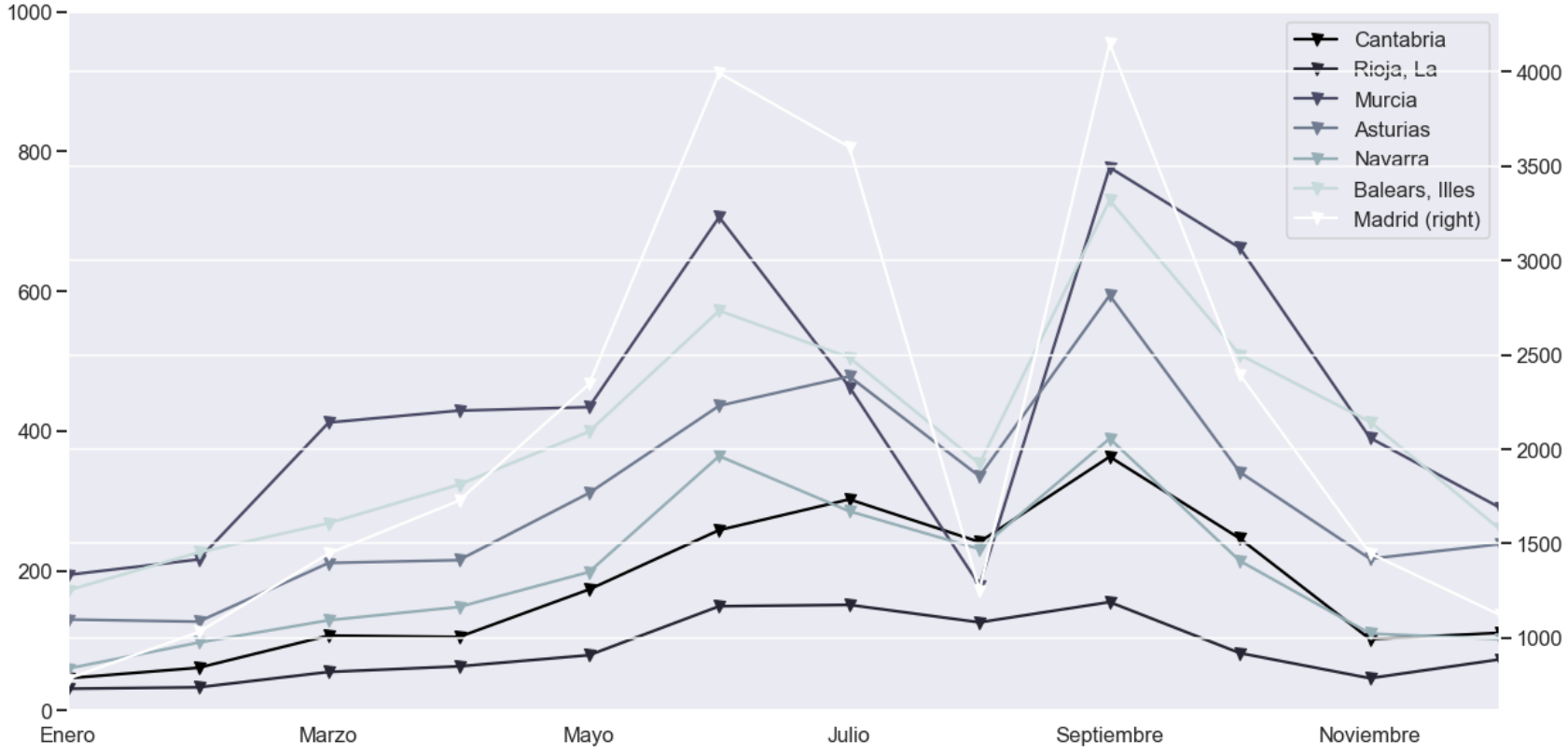
9. `xlim, ylim`: tuplas `(lo, hi)` para delimitar visualización



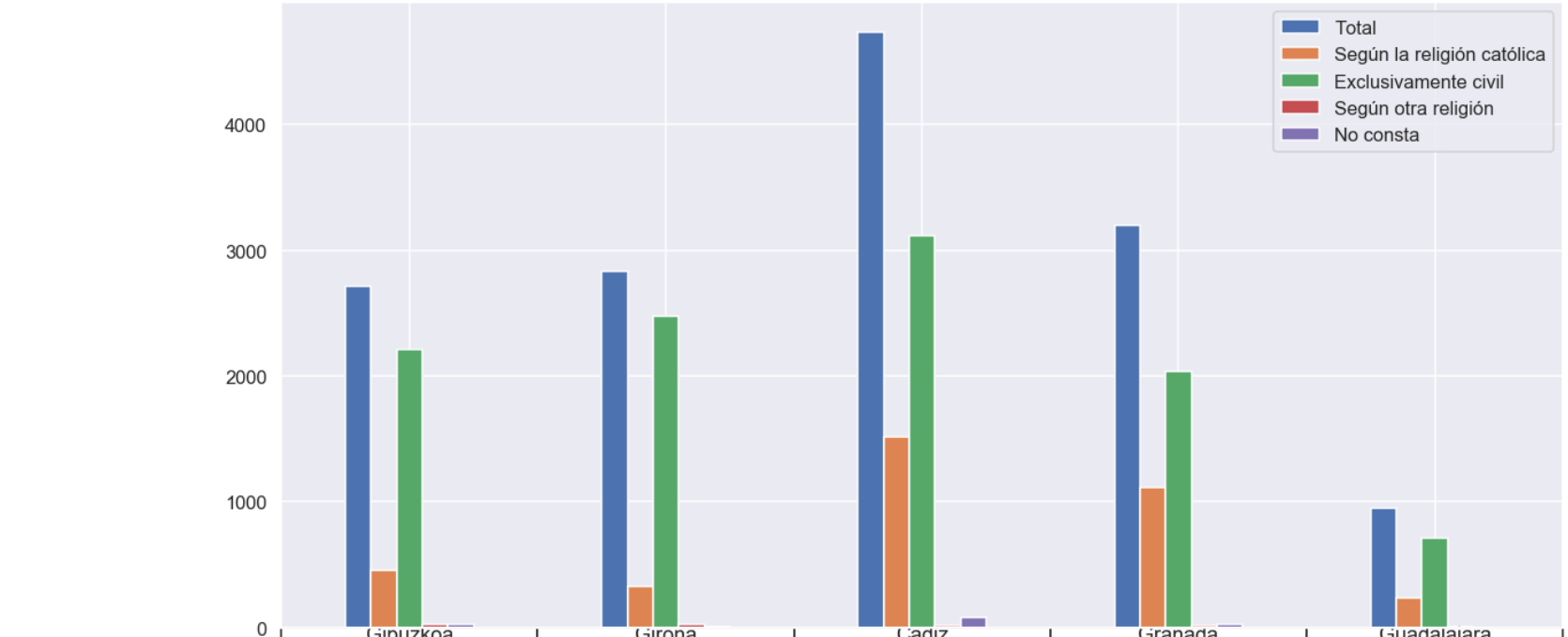
10. `colormap`: mapa de colores (`matplotlib.cm.`)



11. `secondary_y`: segundo eje de ordenadas

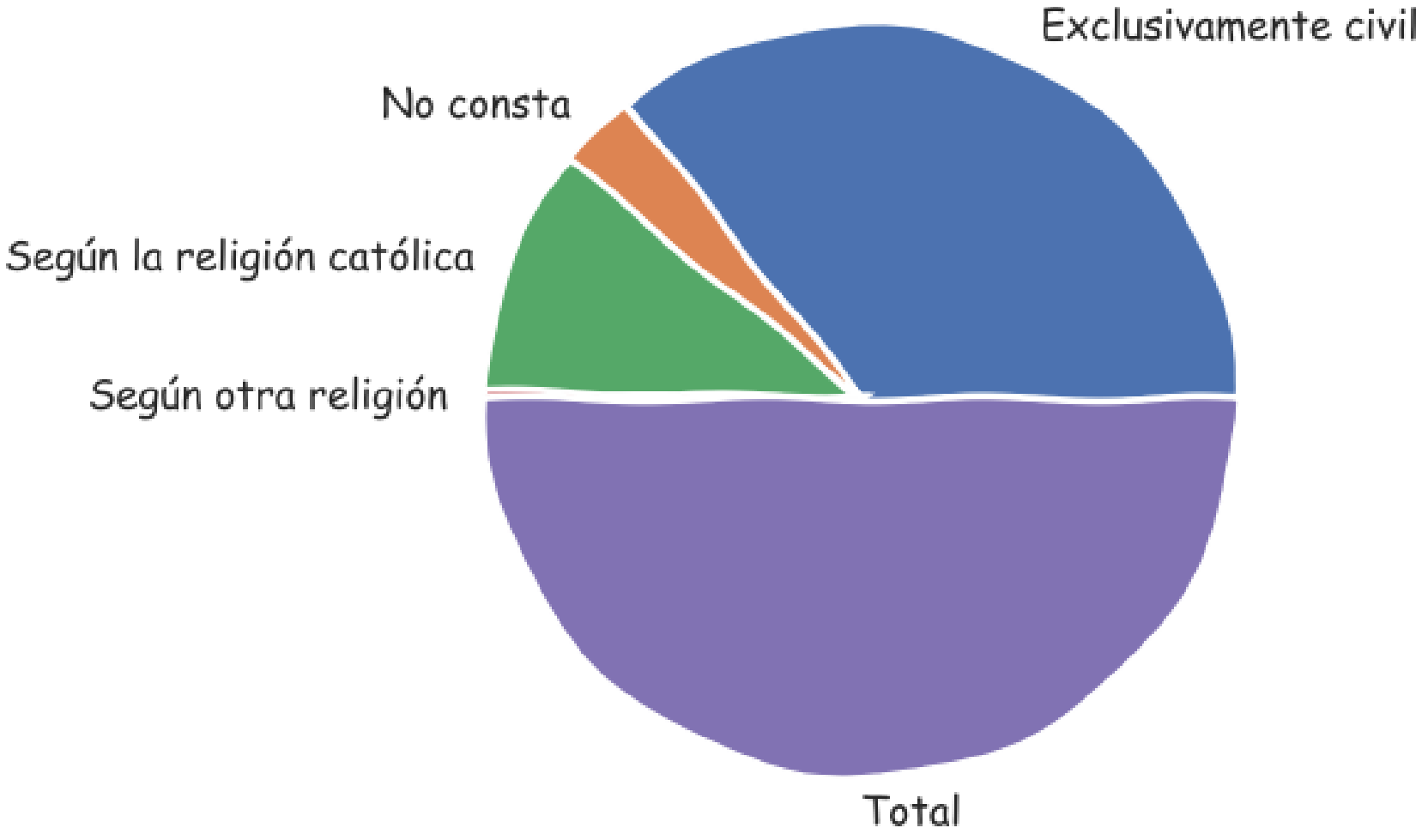


12. `table [True|False]`: mostrar tabla bajo el gráfico



	Gipuzkoa	Girona	Cadiz	Granada	Guadalajara
Total	2719 0	2833 0	4735 0	3198 0	950 0
Según la religión católica	459 0	328 0	1518 0	1116 0	233 0
Exclusivamente civil	2213 0	2477 0	3118 0	2036 0	708 0
Según otra religión	23 0	23 0	17 0	17 0	7 0
No consta	24 0	5 0	82 0	29 0	2 0

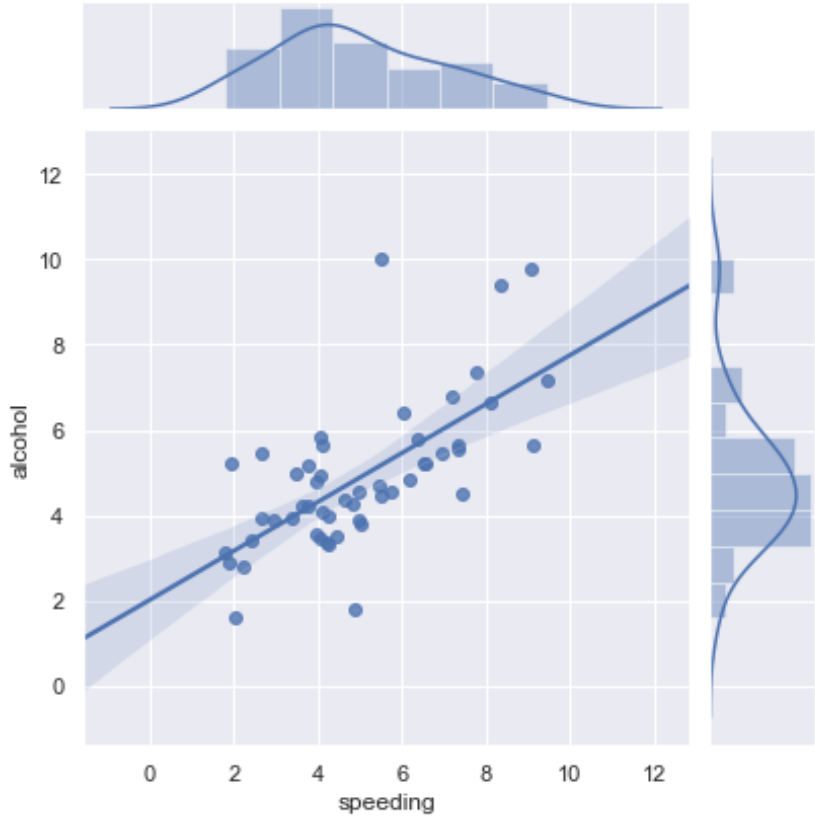
BONUS Generar un gráfico circular con estilo `xkcd`



Alternativas: `seaborn`

Simplifica (aún más) ciertas tareas de visualización

```
sns.set()  # tema por defecto
crashes = sns.load_dataset("ca

with sns.color_palette("husl",
    sns.jointplot(
        "speeding",
        "alcohol",
        crashes,
        kind='reg'
    )
```



Con matplotlib: ...

```
from scipy.stats import gaussian_kde

ax1 = plt.subplot2grid((4, 4), (1, 0), colspan=3, rowspan=3
ax2 = plt.subplot2grid((4, 4), (0, 0), colspan=3)
ax3 = plt.subplot2grid((4, 4), (1, 3), rowspan=3)

crashes.plot.kde(y='speeding', ax=ax2, sharex=ax1, legend=N
crashes.plot.hist(y='speeding', bins=6, ax=ax2, sharex=ax1,
    legend=None, alpha=.5, color='red')
crashes.plot.scatter(x='speeding', y='alcohol', ax=ax1, col

ax2.set_ylabel('')
ax2.set_yticks=[]
ax2.set_yticklabels=[]

# No está soportado directamente el rotado en kde
kde_speeding = gaussian_kde(crashes.alcohol)
y = np.linspace(np.amin(crashes.alcohol), np.amax(crashes.a
```

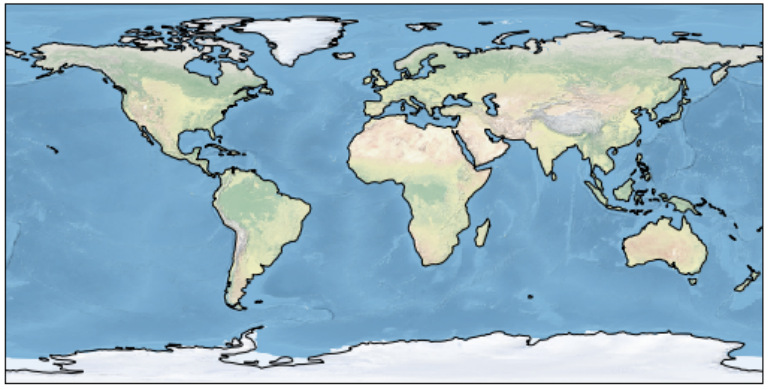
- Hexbin

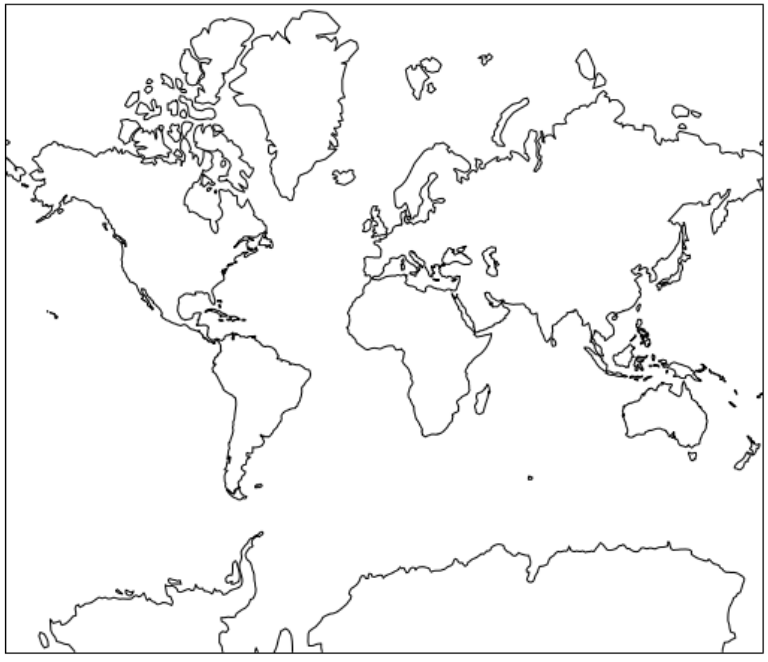
```
x, y = np.random.multivariate_normal(mean, cov, 1000).T
with sns.axes_style("white"):
    sns.jointplot(x=x, y=y, kind="hex", color="k");
```

+ ejemplos

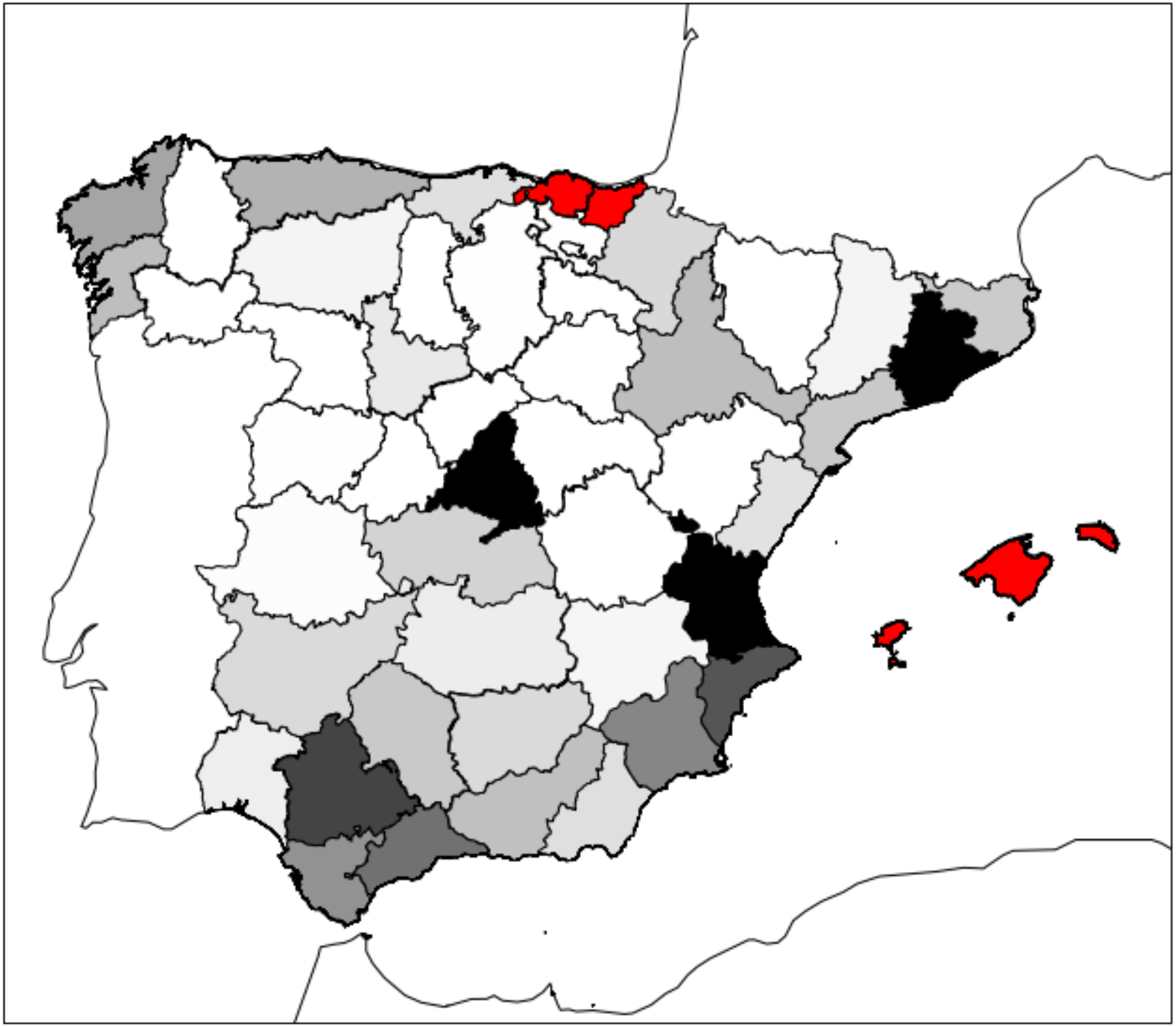
Geovisualización

- Librería standard *de facto*: matplotlib-toolkit (**obsoleta**)
- Oficialmente reemplazada por cartopy
- Alternativa: folium





Número de matrimonios por provincia



Folium

ejemplos online

Cuadernos `jupyter`

- matplotlib/pyplot
- pandas
- caso práctico
- **Soluciones** caso práctico