Clase 037 — Matplotlib: subplots y gridspec

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 4 § 4.6 Multiple Subplots. · Duración estimada: 60 min.

Clase 037 — Matplotlib: subplots y gridspec

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 4 § 4.6 Multiple Subplots. · Duración estimada: 60 min.*

Objetivo

Que el alumno organice múltiples plots en una sola figura — con plt.subplots(n, m) para grillas regulares y con GridSpec para layouts irregulares (un plot grande + varios pequeños). Crítico para informes y dashboards.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Crear grillas regulares con fig, axes = plt.subplots(2, 3, figsize=...).
2. Iterar sobre axes.flat para llenar la grilla con loops.
3. Compartir ejes con sharex=True, sharey=True para comparar.
4. Usar GridSpec para layouts irregulares (1 grande + 3 pequeños).
5. Usar constrained_layout=True en vez de tight_layout() (más confiable).

Temas

#	Tema	Por qué importa
1	plt.subplots(nrows, ncols)	Grilla regular.
2	Iterar con .flat	Llenar muchos plots en loop.
3	sharex/sharey	Comparar con misma escala.
4	GridSpec para layouts irregulares	1 grande + N pequeños.
5	constrained_layout vs tight_layout	El primero es mejor.
6	add_subplot con posiciones custom	Cuando necesitas full control.

Definiciones y características

plt.subplots(n, m)

: Crea figure + grilla regular de n filas × m columnas de axes. Devuelve (fig, axes) donde axes es array 2D — itera con .flat para llenar.

sharex / sharey

: Comparten escala entre axes del grid. Ideal para comparar distribuciones de la misma magnitud sin distorsión visual.

GridSpec

: Layout irregular avanzado: define grilla y asigna spans manualmente ("plot grande arriba + 3 pequeños abajo"). Mucho más flexible que subplots(n, m).

add_subplot(spec)

: Añade un axes a una figure según una posición específica (índice 121, o GridSpec). Útil cuando subplots no alcanza.

constrained_layout=True
: Algoritmo moderno (default en pandas 3+) que ajusta spacing automáticamente. Maneja colorbars, leyendas externas, suptitle sin overlap. Recomendado sobre tight_layout().

Dataset / recursos

Palmer Penguins. Sin descarga.

Ejercicios

1. Grilla 2×2. 4 histogramas de las 4 features numéricas de penguins en una figura.

2. Grilla con loop. Itera axes.flat para plot consistente.

3. sharey=True. 3 boxplots por species lado a lado con misma escala Y.

4. GridSpec irregular. Un scatter grande (2×2) + 1 hist arriba (1×2) + 1 hist a la derecha (2×1) — marginal histograms.

5. constrained_layout. Compara una figura compleja con tight_layout() vs constrained_layout=True — observa diferencia.

Homework verificable

Notebook con penguins: (a) grilla 2×2 hists; (b) 3 boxplots con sharey; (c) layout GridSpec con scatter central + marginales arriba/derecha; (d) misma figura comparando tight_layout vs constrained_layout.

Criterio de aceptación: Sin superposición de labels. Layouts limpios. Marginales alineadas al scatter.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
axes es array 1D cuando esperaba 2D	Si pides subplots(1, 3) o subplots(3, 1),
subplots(1, 1) devuelve axes escalar	No es array. Fix: si quieres array siempre
Plots se superponen / labels cortados	Sin constrained_layout=True ni tight_layou
sharey=True pero un plot tiene escala dist	Quizás esa subplot necesita un eje secunda
GridSpec con índices confusos	El orden es gs[fila, col] igual que NumPy.

Preguntas frecuentes

¿Cuándo subplots(n, m) vs GridSpec?

Regular: subplots. Irregular (un grande + varios pequeños, joint plot, dashboard): GridSpec.

¿fig.suptitle o ax.set_title con subplots?

suptitle = título de toda la figura (encima de todos). ax.set_title = título por subplot. Combinables.

¿Cómo controlo espacio entre subplots?

subplots(..., gridspec_kw={'hspace': 0.3, 'wspace': 0.3}) (ratios relativos al tamaño del axes). Con constrained_layout suele ser automático.

¿figsize cómo elegir?

Para artículos: ancho de columna (típicamente 3.5" para 1 col, 7" para ancho página). Para presentaciones: aspect ratio 16:9 → (12, 6.75).

¿axes.flat o axes.flatten()?

flat es iterador (no copia). flatten() devuelve nuevo array. Para iterar, flat ahorra memoria.

Referencias

- VanderPlas, cap. 4 § 4.6.
- GridSpec tutorial

Siguiente clase

Clase 038 — Matplotlib: legends, colorbars, ticks, anotaciones

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
rng = np.random.default_rng(42)

N = 200
df = pd.DataFrame({
    'x'   : rng.normal(0, 1, N),
    'y'   : rng.normal(0, 1, N),
    'mass': rng.uniform(3000, 5000, N),
    'bill': rng.uniform(35, 50, N),
})
```

Archivos complementarios

- notebook.ipynb