Proyecto de simulación de procesos estocásticos

El proyecto final reúne todo el curso en un caso de riesgo de crédito. Una cartera de préstamos se revisa cada mes y cada préstamo está en uno de cinco estados: **Corriente**, **Atraso30**, **Atraso60**, **Liquidado** e **Incumplimiento**. Los dos últimos son absorbentes. Tu trabajo es estimar la cadena a partir del historial, validarla, calcular la probabilidad de incumplimiento y la duración esperada de un préstamo, y evaluar una política de cobranza antes de aplicarla.

Qué debes entregar

Un notebook que corra de principio a fin sin errores y que incluya:

- **La matriz estimada** desde el historial, con su grafo (`plot_graph`) y la clasificación de estados: clases, transitorios y absorbentes.
- **La validación del modelo:** prueba de propiedad de Markov, prueba de homogeneidad entre la primera y la segunda mitad de la cartera, y la comparación de lo que predice el modelo contra lo observado.
- **Las métricas de absorción:** probabilidad de incumplimiento y meses esperados hasta cerrar el préstamo, desde cada estado transitorio, con PyDTMC y verificadas con una simulación de al menos 10,000 préstamos.
- **Un escenario de cobranza:** qué pasa si una gestión temprana lleva de Atraso30 a Corriente al 55% en lugar del 45% y reduce el paso a Atraso60 del 20% al 10%.
- **Conclusiones:** tres a cinco frases, cada una con un número, y una frase sobre las limitaciones del modelo.

Datos

Los datos se generan con la función de abajo, con semilla fija, así que tus números deben coincidir con los de referencia. Cada préstamo es una lista de estados mensuales que empieza en Corriente y termina en un estado absorbente. Trata la matriz `real` como desconocida: solo la usas para generar el historial.

```
import numpy as np
from pydtmc import MarkovChain

E = ["Corriente", "Atraso30", "Atraso60",
     "Liquidado", "Incumplimiento"]
real = MarkovChain([[0.85, 0.08, 0.00, 0.07, 0.00],
```

```
                    [0.45, 0.30, 0.20, 0.05, 0.00],
                    [0.15, 0.20, 0.30, 0.00, 0.35],
                    [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]], E)

def generar_prestamos(n=2000, seed=42):
    rng, P, out = np.random.default_rng(seed), real.p, []
    for _ in range(n):
        s = [0]
        while s[-1] < 3:
            s.append(int(rng.choice(5, p=P[s[-1]])))
        out.append([E[i] for i in s])
    return out
```

Estimar con varias trayectorias

`fit_sequence` recibe una sola secuencia. Si concatenas los 2,000 préstamos, creas transiciones falsas de Liquidado o Incumplimiento hacia Corriente, y la cadena deja de ser absorbente. En su lugar, cuenta las transiciones dentro de cada préstamo, divide cada fila entre su total y fija en 1 la diagonal de los absorbentes:

```
historial = generar_prestamos()
conteos = np.zeros((5, 5))
for prestamo in historial:
    for a, b in zip(prestamo[:-1], prestamo[1:]):
        conteos[E.index(a), E.index(b)] += 1
conteos[3, 3] = conteos[4, 4] = 1
P_hat = conteos / conteos.sum(axis=1, keepdims=True)
modelo = MarkovChain(P_hat, E)
print(modelo.absorption_probabilities().round(3))
print(modelo.mean_absorption_times().round(2))
```

Valores de referencia

Con la semilla 42 debes obtener, para un préstamo que empieza en Corriente: probabilidad de incumplimiento 0.143 y 13.01 meses esperados hasta cerrar. En el historial, 287 de 2,000 préstamos (14.35%) terminaron en incumplimiento y la duración promedio fue de 13.0 meses, con un percentil 90 de 28 meses. Con la política de cobranza, la probabilidad de incumplimiento desde Corriente baja a 0.075.

Criterios de evaluación

- **Estimación:** matriz construida sin transiciones falsas entre préstamos, filas que suman 1 y estructura correcta (dos estados absorbentes, tres transitorios).
- **Validación:** pruebas de hipótesis interpretadas correctamente y comparación modelo contra observado.
- **Absorción:** t y B calculados con PyDTMC y verificados por simulación, con la diferencia reportada.
- **Escenario:** cambio aplicado solo a la fila de Atraso30, con filas que siguen sumando 1, e impacto expresado en puntos porcentuales y en meses.
- **Reproducibilidad:** semillas fijas y el notebook corre completo tras reiniciar el kernel.

Errores comunes en la entrega

- Concatenar las trayectorias antes de estimar.
- Leer `absorption_probabilities()` como si las filas fueran los estados transitorios; en PyDTMC las filas son los absorbentes.
- Modificar la fila del escenario sin ajustar las demás probabilidades, así que la fila deja de sumar 1 y el constructor lanza `ValidationError`.
- Reportar solo la media del tiempo de absorción cuando la distribución tiene una cola larga.
- Conclusiones sin la limitación principal: el modelo supone que la matriz no cambia con la antigüedad del préstamo ni con el ciclo económico.

Para ir más allá

Agrega un costo por mes en cada estado de atraso y calcula el costo esperado de cobranza por préstamo con la matriz fundamental. O divide la cartera en dos segmentos con matrices distintas y usa `assess_homogeneity` para comprobar si de verdad conviene modelarlos por separado.