Clase 175 — Distribuciones: normal, binomial, Poisson, exponencial

Parte: 3 — Estadística Inferencial y Causal · Fuente: ISLP, cap. 2 + Bruce & Bruce, cap. 2 Data and Sampling Distributions. Duración estimada: 70 min.

Clase 175 — Distribuciones: normal, binomial, Poisson, exponencial

> *Parte: 3 — Estadística Inferencial y Causal · Fuente: ISLP, cap. 2 + Bruce & Bruce, cap. 2 Data and Sampling Distributions. Duración estimada: 70 min.*

Objetivo

Reconocer las cuatro distribuciones de probabilidad que aparecen en el 90 % de los problemas reales de data science —normal, binomial, Poisson, exponencial— sabiendo qué fenómeno modela cada una, cuáles son sus parámetros, cómo simularlas con scipy.stats / numpy.random, y cómo verificar empíricamente si los datos realmente siguen esa distribución antes de aplicar un test que la asuma.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Identificar la distribución apropiada para un fenómeno descrito en lenguaje natural (conteos raros → Poisson, éxitos/fracasos → binomial, tiempos entre eventos → exponencial, sumas/promedios → normal por TCL).
- Calcular media, varianza y cuantiles teóricos con scipy.stats.{norm, binom, poisson, expon} (.mean(), .var(), .ppf(), .pdf()/.pmf()).
- Simular muestras con rng = np.random.default_rng(seed) y comparar histograma vs PDF/PMF teórica.
- Aplicar un Q-Q plot (scipy.stats.probplot) y un Kolmogorov-Smirnov (scipy.stats.kstest) para validar normalidad.
- Reconocer cuándo el Teorema Central del Límite justifica usar normal aunque los datos crudos no lo sean.

Temas

#	Tema	Por qué importa
1	Distribución normal N(μ, σ²)	Base de t-test, ANOVA, intervalos de confi
2	Distribución binomial Bin(n, p)	Conversiones A/B, click-through rate, prop
3	Distribución de Poisson Poi(λ)	Eventos raros por unidad de tiempo/área (f
4	Distribución exponencial Exp(λ)	Tiempos entre eventos Poisson (churn, time
5	Teorema Central del Límite (TCL)	Por qué la normal aparece aunque los datos
6	Verificación empírica: Q-Q plot + KS test	Antes de asumir, mirá.

Definiciones y características

- PDF (Probability Density Function): para variables continuas. f(x) no es probabilidad; la probabilidad es ∫ f(x) dx sobre un intervalo. f(x) puede ser > 1.
- PMF (Probability Mass Function): para variables discretas. P(X = k) directo. Siempre en [0, 1].
- CDF (Cumulative Distribution Function) F(x) = P(X ≤ x): en scipy .cdf(). Su inversa es .ppf() (quantile

function), útil para construir intervalos.
- Normal N(μ, σ²): simétrica, soporte en . Regla 68-95-99.7 (1σ, 2σ, 3σ). Es la única distribución cuya suma de variables independientes sigue siendo de la misma familia exacta.
- Binomial Bin(n, p): suma de n ensayos Bernoulli independientes con éxito p. E[X]=np, Var[X]=np(1-p). Para n grande y p no extrema, ≈ N(np, np(1-p)) — esto es lo que justifica los z-tests de proporciones.
- Poisson Poi(λ): conteo de eventos raros independientes en un intervalo fijo. E[X]=Var[X]=λ (¡equidispersión!). Si tus datos tienen Var/Mean 1, hay sobre-dispersión y Poisson no aplica — considerar binomial negativa.
- Exponencial Exp(λ): tiempo entre eventos Poisson. Memoryless: P(X > s+t | X > s) = P(X > t). Por eso modela mal cosas con desgaste (un motor de 10 años no falla igual que uno nuevo).
- TCL: si X_1, …, X_n son i.i.d. con media μ y varianza finita σ², entonces (X̄ - μ) / (σ/√n) → N(0, 1) cuando n → ∞. Regla práctica: n ≥ 30 alcanza, salvo distribuciones muy asimétricas.
- Q-Q plot: scatter de cuantiles muestrales vs cuantiles teóricos. Si los puntos caen sobre la diagonal, los datos siguen la distribución.

Dataset / recursos

- Conteo de llamados a un call center por hora (sintético): rng.poisson(lam=4.2, size=10_000) → Poisson.
- Datos reales: seaborn.load_dataset('tips') para chequear normalidad de total_bill (no es normal, asimétrico positivo — buen contraejemplo).
- Librerías: numpy, scipy.stats, matplotlib, seaborn.

Ejercicios

1. Simulación y PDF/PMF: con rng = np.random.default_rng(42), generá 10 000 muestras de cada una de las 4 distribuciones con parámetros razonables. Para cada una: histograma con density=True superpuesto con la PDF/PMF teórica de scipy.stats.
2. Cuantiles: calculá scipy.stats.norm(loc=100, scale=15).ppf([0.025, 0.5, 0.975]) (IQ test → IC 95 % poblacional) y verificá que el 2.5 % y 97.5 % muestrales de una simulación con n=100_000 se acerquen.
3. TCL en acción: tomá Exp(λ=1) (claramente no normal). Generá 5 000 promedios de tamaños n {1, 5, 30, 100} y graficá los 4 histogramas lado a lado. Verificá cómo se va volviendo simétrico y campaniforme.
4. Q-Q plot: scipy.stats.probplot(tips.total_bill, dist='norm', plot=plt). Anotá qué muestra el extremo derecho (asimetría positiva → cola larga arriba de la diagonal).
5. ¿Poisson o no?: con los conteos por hora del dataset sintético, calculá mean() y var(). Si var/mean [0.8, 1.2], equidispersión → Poisson plausible. Probá con lam=4.2 (deberías ver ratio ≈ 1) y con datos contaminados (mezclá con rng.poisson(20, size=200) para ver overdispersión).

Homework verificable

Notebook que:

1. Carga tips.total_bill de seaborn.
2. Genera un Q-Q plot contra 'norm' y otro contra 'lognorm'.
3. Aplica scipy.stats.kstest contra ambas (estandarizando los datos).
4. Concluye por escrito qué distribución modela mejor total_bill y por qué (≤ 3 líneas).

Criterio de aceptación: el p-value del KS contra lognormal debe ser mayor que contra normal, y la conclusión

debe mencionar que total_bill está acotado por abajo en 0 y tiene cola derecha — incompatible con normal.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
kstest da p=0.0 aunque los datos "se ven n	Probablemente no estandarizaste. KS contra
Histograma no coincide con la PDF teórica	Olvidaste density=True en plt.hist. Sin es
Aplico Poisson y la varianza es mucho mayo	Sobre-dispersión: hay heterogeneidad ocult
np.random.seed(42) no me da resultados rep	El estado global está deprecado para análi
Asumo normalidad con n=8 porque "el TCL lo	El TCL es asintótico. Con n chico y distri

Preguntas frecuentes

¿Cuándo uso scipy.stats.norm(loc, scale) vs np.random.normal(mu, sigma)?

Para simular muestras: ambos funcionan, pero rng = np.random.default_rng(seed); rng.normal(...) es el patrón moderno reproducible. Para calcular PDF, CDF, cuantiles: scipy.stats.norm, que es un objeto distribución con todos los métodos.

¿Poisson y binomial con n grande y p chica se parecen?

Sí: si $n \to \infty$ y $p \to 0$ con $np = \lambda$ constante, $Bin(n, p) \to Poi(\lambda)$. Por eso "1 cada 1000" se modela igual con cualquiera de las dos. En la práctica, si $n \geq 100$ y $p \leq 0.05$, son intercambiables.

¿La distribución t-Student es lo mismo que la normal?

No, pero converge. $t(\nu)$ con $\nu \to \infty$ tiende a $N(0,1)$. Para $\nu \geq 30$ son visualmente idénticas. La diferencia importa en muestras chicas (la t tiene colas más pesadas, lo que produce intervalos de confianza más anchos — correcto cuando estimás σ).

¿Mis datos tienen que ser normales para hacer un t-test?

No exactamente. Lo que tiene que ser ≈ normal es la distribución muestral de la media, y por TCL eso pasa con $n \geq 30$ aunque los datos crudos no lo sean. Si $n < 30$ y los datos están sesgados, usá bootstrap (Clase 153) o Mann-Whitney (Clase 150).

¿Por qué exponencial es "sin memoria"?

Porque $P(X > s + t \mid X > s) = P(X > t)$. Aplicado a un servidor que lleva 3 h sin caer: la probabilidad de aguantar otra hora es la misma que la de aguantar una hora desde 0. Para sistemas con desgaste (motores, humanos), usá Weibull o Gamma.

Referencias

- ISLP (James et al.), cap. 2 — Statistical Learning, sección sobre distribuciones.
- Bruce, P. & Bruce, A. Practical Statistics for Data Scientists (2ª ed., O'Reilly), cap. 2 Data and Sampling Distributions.
- scipy.stats reference — objetos norm, binom, poisson, expon.
- numpy.random.Generator — API moderna recomendada desde NumPy 1.17.
- 3Blue1Brown — Why π appears in the normal distribution (intuición visual del TCL).

Siguiente clase

Clase 176 — Test t (una muestra, dos muestras, pareado)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb