Clase 186 — CUPED, sequential testing, always-valid p-values

Parte: 3 — Estadística Inferencial y Causal · Fuente: Deng et al. (2013) CUPED + Howard et al. (2021) always-valid + Kohavi et al. (2020). Duración estimada: 85 min.

Clase 186 — CUPED, sequential testing, always-valid p-values

> *Parte: 3 — Estadística Inferencial y Causal · Fuente: Deng et al. (2013) CUPED + Howard et al. (2021) always-valid + Kohavi et al. (2020). Duración estimada: 85 min.*

Objetivo

Aplicar las 3 técnicas modernas que la industria (Microsoft, Netflix, Booking, Spotify) usa para hacer A/B testing más eficientemente: CUPED (variance reduction con covariable pre-experiment), Sequential Testing (mirar el resultado durante el experimento sin inflar α), y always-valid p-values / confidence sequences (Howard et al. 2021, decisión correcta en cualquier momento).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Implementar CUPED: ajustar Y_cuped = Y - θ·(X - E[X]) con θ = Cov(Y,X)/Var(X).
- Calcular la reducción de varianza esperada: 1 - ρ².
- Configurar Group Sequential Testing con O'Brien-Fleming boundaries.
- Aplicar always-valid CIs con la librería confseq.
- Decidir entre frequentist clásico, GST, y mSPRT según contexto.

Temas

- CUPED math: θ óptimo minimiza varianza.
- Implementación: con Y_pre (mismo usuario en período previo) o X (covariable).
- GST: K looks, boundaries pre-definidas.
- O'Brien-Fleming (gasta poco α al principio) vs Pocock (constante).
- mSPRT: ratio test que provee p-value válido siempre.
- Confidence sequences: IC válido en cualquier t.

Definiciones y características

- CUPED: Controlled-experiment Using Pre-Experiment Data.
- Variance reduction: nuevo Var(Y_cuped) = Var(Y) · (1 - ρ²).
- Sequential testing: tomar decisión antes de N planificado.
- GST: predefine K mira-instantes, pasa α budget según schedule.
- Always-valid p-value: válido bajo cualquier optional stopping rule.
- confseq library: implementación de Howard et al.

Dataset / recursos

- Simulación A/B con numpy.random.default_rng.
- Librerías: numpy, scipy, confseq (pip install confseq).

Ejercicios

1. CUPED implementation: simular X_pre, Y_post = α·X_pre + tratamiento + ε. Calcular θ. Comparar Var(Y) vs Var(Y_cuped).
2. Variance reduction: con ρ=0.7, calcular reducción esperada (= 51 %); verificar con simulación.
3. Peeking inflado: bajo H, simular 1000 experimentos con 5 looks naïve, contar % de rejects. Debería ser ≈ 18 %.
4. O'Brien-Fleming: implementar boundaries con rpy2 + gsDesign (o aproximación). Verificar α controlled.
5. Always-valid CI: confseq.bounds.normal_log_mixture_bound sobre stream simulado. Plotear CI a lo largo del tiempo.

Homework verificable

Comparar 4 enfoques sobre simulación A/B realista:

1. Frequentist clásico (fixed N).
2. CUPED + frequentist.
3. Naïve peeking cada 1k samples (α inflated).
4. Always-valid (confseq).

Reportar para cada uno: tipo I error (bajo H), poder (bajo H), promedio sample size hasta decisión.

Criterio de aceptación: CUPED reduce sample requerido ~30 % con poder igual; naïve peeking infla α a 0.15-0.25; always-valid mantiene α=0.05 con +20 % sample.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
CUPED reduce varianza solo si ρ > 0	Si X no correlaciona, no ayuda. Fix: elegi
Peeking sin corrección	α inflado. Fix: GST o always-valid.
GST con boundaries mal calibradas	Engineers re-implementan mal. Fix: librerí
Always-valid muy conservador con pocos sam	Inherente. Fix: aceptarlo o usar GST.
Reportar GST como "p-value normal"	Distinto significado. Fix: documentar que

Preguntas frecuentes

CUPED siempre vale la pena?

Sí, si tenés X correlated y es free de computar. Microsoft reporta 50 % menos samples en muchos casos.

GST o always-valid?

GST: K looks fijos, simple, comunidad estadística. Always-valid: mirá cuando querás, más conservador. Para casos modernos (peeking continuo), always-valid.

Implementations open source?

- GST: gsDesign (R), pyabtest o reimplementación.
- Always-valid: confseq (Python), cpsequential (R).

Bayesian A/B con stopping?

También válido para optional stopping, pero requiere prior honesto. Decisión: P(B > A | data) > 0.95.

Industria realmente lo usa?

Sí: Microsoft (CUPED), Netflix (sequential), Optimizely (always-valid). Buscar "Trustworthy Online Controlled Experiments" book.

Referencias

- Deng, Xu, Kohavi & Walker (2013), Improving the Sensitivity of Online Controlled Experiments by Utilizing Pre-Experiment Data, WSDM.
- Howard, Ramdas, McAuliffe & Sekhon (2021), Time-uniform, nonparametric, nonasymptotic confidence sequences, Annals of Statistics.
- Kohavi, Tang & Xu (2020), Trustworthy Online Controlled Experiments, Cambridge.
- confseq.

Siguiente clase

Clase 187 — Diseño experimental

Apéndice: notebook (primer bloque)

Tres trucos modernos de A/B testing industrial: CUPED reduce varianza usando pre-experiment data (~50% menos sample). Sequential testing te deja peekear sin inflar el error tipo I. Requiere: pip install numpy scipy.

```
import numpy as np
from scipy import stats

rng = np.random.default_rng(42)
n = 10_000
# Pre-experiment covariate (X), correlación ~0.7 con outcome
X_ctrl = rng.normal(10, 3, n)
X_trt  = rng.normal(10, 3, n)
noise  = lambda k: rng.normal(0, 2.1, k)  # ruido idiosincratico
Y_ctrl = X_ctrl + noise(n)            # mean ~ 10
Y_trt  = X_trt + noise(n) + 0.5       # treatment effect = 0.5
print(f'mean ctrl = {Y_ctrl.mean():.3f}  mean trt = {Y_trt.mean():.3f}  diff = {Y_trt.mean()-Y_ctrl.mean():.3f}')
```

Archivos complementarios

- notebook.ipynb