 QBank Scaffold v6: Full Sympson–Hetter iterative calibration
- Offline simulator + iterative SH estimator that targets per-item exposure cap τ
- Admin endpoint to trigger calibration (with dry-run + parameters)
- Updates selection service to count "proposals" (instrumentation hook)
#
Output: /mnt/data/qbank_scaffold_v6.zip

import os, zipfile, pathlib, textwrap, json

ROOT = "/mnt/data/qbank_scaffold_v6"
os.makedirs(ROOT, exist_ok=True)

def write(path, content):
    os.makedirs(os.path.dirname(path), exist_ok=True)
    with open(path, "w", encoding="utf-8") as f:
        f.write(textwrap.dedent(content))

---------- README ----------
write(f"{ROOT}/README.md", """
QBank Scaffold (v6) — Full Sympson–Hetter (S–H) iterative calibration

This drop wires a **true S–H iterative estimator** that simulates CAT sessions and updates
per-item acceptance probabilities `sh_p` so the **probability an examinee sees an item**
(i.e., exposure rate) does not exceed τ.

What's included:
- `analytics/calibration/sh_iterative.py`: Iterative S–H with simulation → updates `item_exposure_control`.
- Admin API: `POST /v1/admin/exposure/calibrate_sh` to run (or dry-run) calibration on demand.
- Selector instrumentation: a hook to count **proposals** (times an item is proposed before acceptance).

Quick run (dev-safe defaults):
```bash
python analytics/calibration/sh_iterative.py \
  --dsn postgresql://qbank:qbank@localhost:5432/qbank \
  --exam DEMO-EXAM --tau 0.2 --n 500 --len 30 --iters 6 --alpha 0.6
API trigger (synchronous; use low n/iters in prod or run offline via cron/job):

bash
Always show details
```

Copy
curl -X POST http://localhost:8000/v1/admin/exposure/calibrate_sh \
  -H "Authorization: Bearer <ADMIN_JWT>" -H "Content-Type: application/json" \
  -d '{"exam_code":"DEMO-
EXAM","tau":0.2,"n":400,"test_len":25,"iters":5,"alpha":0.6,"dry_run":false}'
Notes:

We estimate exposure as the share of examinees who received the item at least once.

Update rule (damped): k_i <- clamp(floor, ceil, k_i * (tau / max(r_i, eps))^alpha), where r_i is observed exposure rate under the current k; alpha∈(0,1] damps oscillations.

If an item is chronically under-exposed (r_i ≪ τ), k_i will drift up toward 1; if over-exposed,
k_i reduces multiplicatively.
""")

---------- Simulator & Iterative SH ----------
write(f"{ROOT}/analytics/calibration/sh_iterative.py", """
import argparse, math, random, statistics
import psycopg2, psycopg2.extras

D = 1.7

def logistic(x: float) -> float: return 1.0 / (1.0 + math.exp(-x))
def prob_3pl(theta: float, a: float, b: float, c: float) -> float:
return c + (1.0 - c) * logistic(D * a * (theta - b))
def fisher_info_3pl(theta: float, a: float, b: float, c: float) -> float:
P = prob_3pl(theta, a, b, c); Q = 1.0 - P
if P<=0 or Q<=0 or (1.0-c)<=0: return 0.0
return (D2)*(a2)(Q/P)((P-c)/(1.0-c))**2

def sample_theta(dist: str = "normal0,1"):
# extensible: normal0,1 | uniform-1,1 | fixed0
if dist.startswith("normal"):
mu, sd = 0.0, 1.0
return random.gauss(mu, sd)
if dist.startswith("uniform"):
return random.uniform(-1.0, 1.0)
return 0.0

```
def load_pool(conn, exam_code: str):
sql = '''
SELECT qv.question_id, qv.version, qv.topic_id,
COALESCE(ic.a, 1.0) AS a, COALESCE(ic.b, 0.0) AS b, COALESCE(ic.c, 0.2) AS c,
COALESCE(iec.sh_p, 1.0) AS sh_p
FROM question_publications qp
JOIN question_versions qv ON qv.question_id=qp.question_id AND
qv.version=qp.live_version
LEFT JOIN item_calibration ic ON ic.question_id=qv.question_id AND
ic.version=qv.version AND ic.model='3pl'
LEFT JOIN item_exposure_control iec ON iec.question_id=qv.question_id AND
iec.version=qv.version
WHERE qp.exam_code=%s AND qv.state='published'
'''
with conn.cursor(cursor_factory=psycopg2.extras.DictCursor) as cur:
cur.execute(sql, (exam_code,))
rows = cur.fetchall()
items = []
for r in rows:
items.append({
"qid": int(r["question_id"]), "ver": int(r["version"]),
"topic": r["topic_id"], "a": float(r["a"]), "b": float(r["b"]), "c": float(r["c"]),
"k": float(r["sh_p"]) if r["sh_p"] is not None else 1.0
})
return items

def simulate_once(pool, n: int, test_len: int, theta_dist: str, use_k: bool = True,
seed: int | None = None):
if seed is not None: random.seed(seed)
# Tallies:
# seen[i_key] = number of examinees who saw item at least once (exposure rate
numerator)
# proposed[i_key] = number of times item was proposed (instrumentation; unused
by update but exposed for analysis)
seen = { (it["qid"], it["ver"]): 0 for it in pool }
proposed = { (it["qid"], it["ver"]): 0 for it in pool }

python
Always show details

Copy
for _ in range(n):
    theta = sample_theta(theta_dist)
```

```
    administered_keys = set()
    for _pos in range(test_len):
        # rank by Fisher information for this theta among items not yet seen in this
test
        candidates = [it for it in pool if (it["qid"], it["ver"]) not in administered_keys]
        if not candidates: break
        scored = [(fisher_info_3pl(theta, it["a"], it["b"], it["c"]), it) for it in
candidates]
        scored×sort(key=lambda x: x[0], reverse=True)

        chosen = None
        for _, it in scored:
            key = (it["qid"], it["ver"])
            proposed[key] += 1
            if use_k:
                if random.random() <= max(0.0, min(1.0, it["k"])):
                    chosen = it; break
                else:
                    continue
            else:
                chosen = it; break
        if chosen is None:  # if everything rejected, force top
            chosen = scored[0][1]
        key = (chosen["qid"], chosen["ver"])
        if key not in administered_keys:
            administered_keys.add(key)

    # update seen tallies (each administered item counts once per examinee)
    for key in administered_keys:
        seen[key] += 1

return seen, proposed
def iterative_sh(pool, tau: float, n: int, test_len: int, iters: int, alpha: float,
theta_dist: str, floor: float, ceil: float, seed: int | None = None):
# copy ks
k = { (it["qid"], it["ver"]): float(it["k"]) for it in pool }
history = []
for t in range(iters):
# write ks back into pool
for it in pool:
it["k"] = k[(it["qid"], it["ver"])]
seen, proposed = simulate_once(pool, n=n, test_len=test_len,
theta_dist=theta_dist, use_k=True, seed=(None if seed is None else seed + t))
# exposure rates: proportion of examinees who saw the item at least once
```

```
r = { key: seen[key] / max(1, n) for key in seen }
# update rule (damped multiplicative adjustment)
newk = {}
diffs = []
for it in pool:
key = (it["qid"], it["ver"])
ri = r×get(key, 0.0)
ki = k[key]
if ri <= 0.0:
newk[key] = min(1.0, max(floor, ki * 1.1)) # nudge up gently if never seen
else:
ratio = tau / ri
adj = ratio ** alpha
cand = ki × adj
newk[key] = min(1.0, max(floor, min(ceil, cand)))
diffs.append(abs(ri - min(tau, ri))) # track magnitude (ri itself relative to cap)
k = newk
avg_exp = statistics×mean(r×values()) if r else 0.0
max_over = max((ri - tau for ri in r.values()), default=0.0)
history.append({"iter": t+1, "avg_exp": avg_exp, "max_over": max_over})
# return final k, last seen/proposed, history
return k, seen, proposed, history

def upsert_k(conn, kmap):
with conn.cursor() as cur:
cur.execute("SET search_path TO public")
for (qid, ver), kval in kmap.items():
cur.execute(
'''INSERT INTO item_exposure_control(question_id, version, sh_p)
VALUES (%s,%s,%s)
ON CONFLICT (question_id,version) DO UPDATE SET sh_p=EXCLUDED.sh_p,
updated_at=now()''',
(qid, ver, float(kval))
)
conn.commit()

def main():
ap = argparse×ArgumentParser()
ap.add_argument("--dsn", required=True)
ap.add_argument("--exam", "--exam_code", dest="exam_code", required=True)
ap.add_argument("--tau", type=float, default=0.2, help="Max exposure per item
(probability)")
ap.add_argument("--n", type=int, default=500, help="Number of examinees to
simulate per iteration")
```

```
ap.add_argument("--len", "--test_len", dest="test_len", type=int, default=30,
help="Test length (items)")
ap.add_argument("--iters", type=int, default=6, help="Number of S–H iterations")
ap.add_argument("--alpha", type=float, default=0.6, help="Damping exponent in
(0,1]")
ap.add_argument("--theta", "--theta_dist", dest="theta_dist",
default="normal0,1")
ap.add_argument("--floor", type=float, default=0.02)
ap.add_argument("--ceil", type=float, default=1.0)
ap.add_argument("--seed", type=int, default=None)
ap.add_argument("--dry_run", action="store_true")
args = ap×parse_args()

lua
Always show details

Copy
conn = psycopg2.connect(args×dsn)
pool = load_pool(conn, args.exam_code)
if not pool:
    print("No items found for exam", args.exam_code); return

kmap, seen, proposed, hist = iterative_sh(
    pool, tau=args.tau, n=args.n, test_len=args.test_len, iters=args.iters,
    alpha=args.alpha, theta_dist=args×theta_dist, floor=args.floor, ceil=args.ceil,
seed=args×seed
)

print("Iterations summary:", hist)
top = sorted(((kmap[(it["qid"], it["ver"])], it["qid"], it["ver"]) for it in pool),
reverse=True)[:5]
print("Top k after calibration (first 5):", top[:5])

if not args.dry_run:
    upsert_k(conn, kmap)
    print(f"Upserted {len(kmap)} k-values into item_exposure_control")
conn.close()
if name == "main":
main()
""")

---------- Admin API: trigger calibration ----------
write(f"{ROOT}/qbank-backend/app/api/admin.py", """
from fastapi import APIRouter, Depends, HTTPException
```

```
from pydantic import BaseModel, Field
from typing import List, Optional
from sqlalchemy.orm import Session
from sqlalchemy import text
import os, subprocess, sys
from app.core.database import get_db
from app.core.auth import require_roles
from app.models.orm import ItemExposureControl

router = APIRouter()

class ItemRow(BaseModel):
question_id: int
version: int
topic_id: Optional[int] = None
sh_p: float
recent_attempts: int | None = 0

@router.get("/exposure/items", response_model=List[ItemRow],
dependencies=[Depends(require_roles("admin"))])
def list_items(limit: int = 100, db: Session = Depends(get_db)):
rows = db.execute(text("""
SELECT qv.question_id, qv.version, qv.topic_id,
COALESCE(iec.sh_p, 1.0) as sh_p,
(SELECT count(*) FROM user_responses ur WHERE ur.question_id=qv.question_id
AND ur.version=qv.version AND ur.created_at>now()-interval '7 days') as
recent_attempts
FROM question_versions qv
LEFT JOIN item_exposure_control iec ON iec.question_id=qv.question_id AND
iec.version=qv.version
WHERE qv.state='published'
ORDER BY recent_attempts DESC NULLS LAST
LIMIT :lim
"""), {"lim": limit}).all()
return [ItemRow(question_id=r[0], version=r[1], topic_id=r[2], sh_p=float(r[3]),
recent_attempts=r[4] or 0) for r in rows]

class SetSh(BaseModel):
question_id: int
version: int
sh_p: float

@router.post("/exposure/set", dependencies=[Depends(require_roles("admin"))])
def set_sh(payload: SetSh, db: Session = Depends(get_db)):
```

```
if payload.sh_p < 0 or payload.sh_p > 1: raise HTTPException(400, "sh_p must be in [0,1]")
row = db.get(ItemExposureControl, {"question_id": payload.question_id, "version": payload.version})
if row: row.sh_p = payload.sh_p
else: db.add(ItemExposureControl(question_id=payload.question_id, version=payload×version, sh_p=payload.sh_p))
db.commit()
return {"ok": True}

class CalibrateRequest(BaseModel):
exam_code: str = Field(...)
tau: float = 0.2
n: int = 400
test_len: int = 25
iters: int = 5
alpha: float = 0.6
theta_dist: str = "normal0,1"
floor: float = 0.02
ceil: float = 1.0
dry_run: bool = False

@router.post("/exposure/calibrate_sh", dependencies=[Depends(require_roles("admin"))])
def calibrate_sh(payload: CalibrateRequest):
# This is a simple synchronous wrapper around the CLI; in production, prefer a job runner.
dsn = os.getenv("DATABASE_URL", "postgresql+psycopg2://qbank:qbank@localhost:5432/qbank")
cmd = [
sys.executable, "analytics/calibration/sh_iterative.py",
"--dsn", dsn, "--exam", payload.exam_code,
"--tau", str(payload.tau), "--n", str(payload.n), "--len", str(payload.test_len),
"--iters", str(payload.iters), "--alpha", str(payload.alpha),
"--theta", payload.theta_dist, "--floor", str(payload.floor), "--ceil", str(payload.ceil)
] + (["--dry_run"] if payload.dry_run else [])
try:
proc = subprocess.run(cmd, capture_output=True, text=True, check=True)
return {"ok": True, "stdout": proc.stdout}
except subprocess.CalledProcessError as e:
raise HTTPException(status_code=500, detail=e×stdout + "\n" + e.stderr)
""")
```

```
---------- Selection service: proposal hook (documentation stub) ----------
write(f"{ROOT}/qbank-backend/app/services/proposals.py", """

Placeholder for live instrumentation.
If you want to count proposals in production (item was proposed prior to S–H acceptance),
wire this function from your selector path.
from app.core.cache import redis_client

def bump_proposed(question_id: int, version: int) -> None:
# Daily counter; useful for diagnostics
import datetime as dt
day = dt×datetime×utcnow()×strftime("%Y%m%d")
redis_client.incr(f"proposed:{day}:{question_id}:{version}", 1)
""")

---------- Keep minimal main to include admin router ----------
write(f"{ROOT}/qbank-backend/app/main.py", """
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from app.api.admin import router as admin_router

app = FastAPI(title="QBank API v6 (admin demo)", version="6.0.0")
app.add_middleware(CORSMiddleware, allow_origins=[""],
allow_credentials=True, allow_methods=[""], allow_headers=["*"])
app.include_router(admin_router, prefix="/v1/admin", tags=["admin"])

@app.get("/health")
def health(): return {"status": "ok"}
""")

---------- Zip ----------
zip_path = "/mnt/data/qbank_scaffold_v6.zip"
with zipfile.ZipFile(zip_path, "w", zipfile.ZIP_DEFLATED) as z:
for p in pathlib.Path(ROOT).rglob("*"):
z.write(p, p.relative_to(ROOT))

zip_path

Always show details

Copy
Result
```

'/mnt/data/qbank_scaffold_v6.zip'
Done — full Sympson–Hetter iterative calibration is wired up.

Download v6 (full SH calibration)

What's in v6:

Offline simulator + iterative SH estimator
analytics/calibration/sh_iterative.py simulates CAT sessions and iteratively updates sh_p so each item's exposure (probability an examinee sees the item at least once) stays ≤ τ. It uses a damped multiplicative update to avoid oscillation and writes results to item_exposure_control.

Admin calibration endpoint
POST /v1/admin/exposure/calibrate_sh (sync) wraps the CLI; supports dry_run, tau, n, test_len, iters, alpha, theta_dist, and bounds.

Proposal instrumentation hook
app/services/proposals.py gives you a one-liner to increment daily "proposed" counters if you want live diagnostics.

How to run it quickly:

Simulate and update sh_p offline

bash
Copy
Edit
python analytics/calibration/sh_iterative.py \
  --dsn postgresql://qbank:qbank@localhost:5432/qbank \
  --exam DEMO-EXAM --tau 0.2 --n 500 --len 30 --iters 6 --alpha 0.6
Or via API (synchronous; use smaller n/iters in dev):

bash
Copy
Edit
POST /v1/admin/exposure/calibrate_sh
{
  "exam_code": "DEMO-EXAM",
  "tau": 0.2,
  "n": 400,
  "test_len": 25,
  "iters": 5,
  "alpha": 0.6,

```
  "dry_run": false
}
```

(Optional) Wire live proposal counting

Call bump_proposed(qid, ver) from your selector right before SH acceptance sampling.

Defaults & tuning tips:

tau: typical caps are 0.15–0.25 for large pools; tighten for high-stakes pools.

alpha: 0.5–0.7 damps volatility; raise toward 1.0 if you want faster convergence.

n & test_len: increase for smoother estimates; they scale runtime linearly.

Bounds: floor 0.02 guards against items becoming “dead”; keep ceil at 1.0.