Wholesale power-price analysis pipeline, North America

Jake Tae-moon Jo · Engineering lead · https://jake-jo-portfolio.vercel.app/en/work/grid-constraint-analysis/

Built a pipeline that writes transmission-constraint memos, refined it through 22 hypothesis experiments, and rolled back an adopted one on finding the scoring code was not the code that writes the memos.

6 hypotheses adopted	29→46%	−39 pts	+22 · +13
of 22 (6 rejected, 8 open, 2 adopted as input)	lead-trigger hits, one hypothesis (dev; train 22→32%)	agreement, production path (train)	from the definition split (train · dev)

Results

Over 42 days I logged 22 hypothesis experiments, adopting 6 and rejecting 6; one of the rejections is hypothesis 04, which was merged and then rolled back.

Every client change request was shipped within 24 hours.

From the first commit on August 3 to the last push on September 14, 42 days, I logged 22 hypothesis experiments: 6 adopted and 6 rejected, with 8 open and 2 adopted as input.

Experiment 45 raised the lead-trigger hit rate from 22% to 32% on train and from 29% to 46% on dev.

Scoring with the same function as production exposed the memo agreement at −39 points on train; splitting the outage grade's definition then raised it by 22 points on train and 13 on dev (the effect of a definition change, not of an adopted hypothesis). On the production path critical recall for outages went from 2% to 9%.

I rejected five hypotheses that won on train and lost on dev. Experiment 49 was adopted with a dev net gain of 0.

Whether the traders actually used these CSV memos is not recorded in the

repo. The result stops at what was built and what was measured.

Period

Aug – Sep 2026

Role

Only human on the build team; designed and ran the AI agent team

Team

Me plus AI agents (the only commit author)

Platforms

Command-line analysis pipeline · CSV memos · hypothesis experiment harness

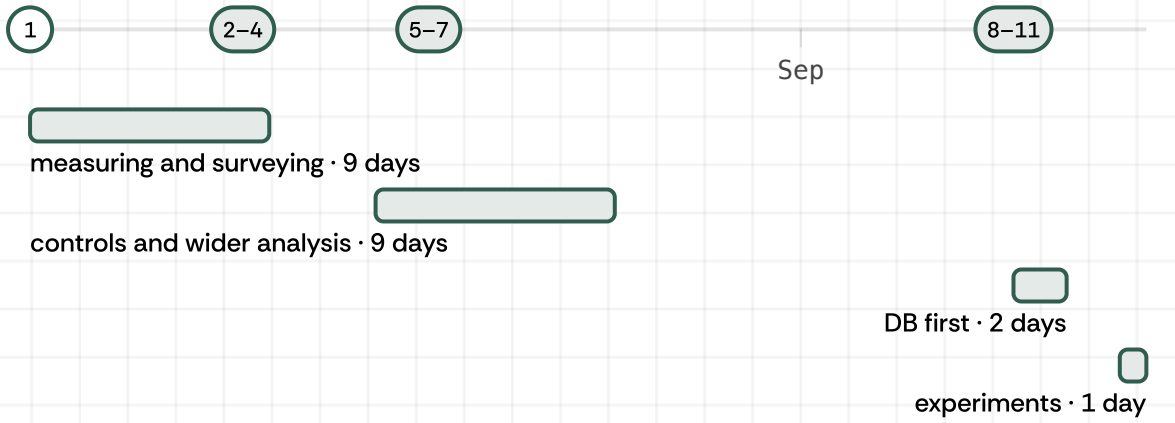
Context

Client work automating the hand analysis traders did to explain why a transmission constraint bound in a North American wholesale power market

42 days and 22 experiments

Over 42 days I logged 22 hypothesis experiments, adopting 6 and rejecting 6; one of the rejections is hypothesis 04, which was merged and then rolled back.

42 days	22	6
first commit to last push	logged experiments	adopted



1	Aug 3, 2026	**First commit**
2	Aug 11, 2026	**Survey of all the traders' memos** 1,482 memos and 4,853 cause lines
3	Aug 12, 2026	**Separate repo for service migration** 89 constraint profiles as the regression baseline (through Aug 13); the service itself not started
4	Aug 17, 2026	**Two CSVs shipped without passing the gate** Steps had been run by hand
5	Aug 18, 2026	**Publish stamp** Publishing needs a stamp whose note fingerprint matches
6	Aug 19, 2026	**One entry point and a step-table check**

		Two-way check between the step table and the script found 4 missing steps
7	Aug 24, 2026	**Blind by default** So the comparison is not circular
8	Sep 9, 2026	**Inputs from the client DB first** One DB for collection by Sep 11
9	Sep 13, 2026	**Hypothesis harness** Pre-register, score on train/dev, verdict; 44 reports across Sep 13–14
10	Sep 14, 2026	**Hypothesis 04 rolled back** Re-measured on the production path and rejected
11	Sep 14, 2026	**Last push** 6 adopted, 6 rejected, 8 open, 2 taken as inputs

Problem and context

The job was to generate, in numbers that can be traced, the "why did it bind" memos that traders wrote by hand.

In North American wholesale power markets, prices split when a transmission constraint binds in the real-time market. The people who watch congestion-price swings analysed "why did it bind" by hand and left a memo. The job was to generate that explanation automatically in numbers that can be traced; the output is a CSV with one row per constraint.

For every constraint the memo had to answer three questions in numbers: which price node moves, in which direction and by how much; whether the cause is load or thermal limits, a transmission outage, renewables, a neighbouring corridor or a border flow; and whether the day-ahead market already knew. What the day-ahead market has already priced is not an edge. Causes are classified in a table of six causes (the last being "no supporting cause") against three day-ahead columns, and "no supporting cause" is not a default: it is what remains after the other causes were tested and all failed.

The reference answers are the traders' past memos, written by hand. On August 11 I surveyed all of them: 1,482 memos and 4,853 cause lines.

The sample was small. Hours bound per constraint have a median of 7, and 0 of 192 exceed 100 hours. On August 28 I diagnosed that regression, ML and hierarchical models were out, leaving only deterministic decomposition, which needs no sample.

From one binding constraint to a memo

One constraint passes 73 calls, an LLM analyse-then-refute step, an independent recomputation and 21 gate checks to become one CSV row, and by design a person puts that row into the system.

1. **Three questions** ☐System
   Which node moves and by how much · what caused it · did the day-ahead market already know
2. **Collect one day** ☐System
   Client DB first, public market and weather feeds as backup; no start without required inputs
3. **Break it down in numbers** ☐System
   Outages, load, renewables, neighbouring corridors and cross-border flow in 73 calls; stops at the first failure
4. **LLM analyst** ◇Agent
   Reviews the draft in 20 parallel lanes; with the next step about 85% of the time
5. **LLM refute** ◇Agent
   Default verdict REFUTED; CONFIRMED needs a cited source
6. **Independent recomputation** ☐System
   Counts as passing only if it reproduces 8 past errors
7. **Gate and stamp** ☐System
   21 checks; a pass writes a stamp with the note fingerprint, a failure deletes it
8. **A person** ○Person
   By design, never written into the system automatically

My role

I set the direction and made the adopt and roll-back calls; the code and the experiment reports were written almost entirely together with AI agents.

I am the commit author of all 433 commits in this repo, and there are no commits, PRs or reviews from another person. 422 of the 433 (97.5%) carry an AI co-author mark, and the author field of all 44 experiment reports points to the AI. There is no record of how the writing was split between me and the AI, so the role is not "wrote it myself" but "designed and ran the AI agent team".

What I decided is recorded in the documents with dates. Outage history never trusts an old cache and is fetched fresh (Aug 17). The analysis

starts only once the required inputs are in (Aug 23). Reference memos are not read by default (Aug 24). The definition of critical and the scoring rule (Sep 2). The pipeline is fixed on small windows first (Sep 7). And three on September 14: take from the client DB first, adopt experiment 16, which switched an input to the DB, and open the holdout once before merging hypothesis 04, then roll it back.

I built four things. The analysis pipeline: from one entry point, a day's collection and 73 calls, two LLM review steps (analyse, refute), an independent recomputation, 21 gate checks and a publish stamp, ending in a CSV. The memo, whose lint fails raw decimal notation. The hypothesis harness, which scores six memo segments separately. And a separate repository holding the pipeline for service migration: its README lists the pipeline at 15 steps, and it uses 89 constraint profiles as a regression baseline. The service itself was not started.

I also set the working rules. Only a person signs in or authenticates, the pipeline never writes into the system's memo field and only produces output, and fixes start on 4–10-case windows. These rules are written down as standing instructions.

Who decided what

I alone set the direction and made the adopt and roll-back calls, 422 of 433 commits carry the AI co-author mark, and publishing is left to the gate and the stamp.

Me, the only person

Aug 23: the analysis starts only once the required inputs are in
Aug 24: blind by default
Sep 2: critical means a surprise to the day-ahead market; more verdicts at 52% accuracy
Sep 7: fix the pipeline on 4–10-case windows first
Sep 14: opened the holdout once, merged hypothesis 04, then rolled it back

AI agent team

AI co-author mark on 422 of 433 commits (97.5%)
The author field of all 44 experiment reports points to the AI
Analyse and refute per constraint, 20 parallel lanes
No record of the human-to-AI split

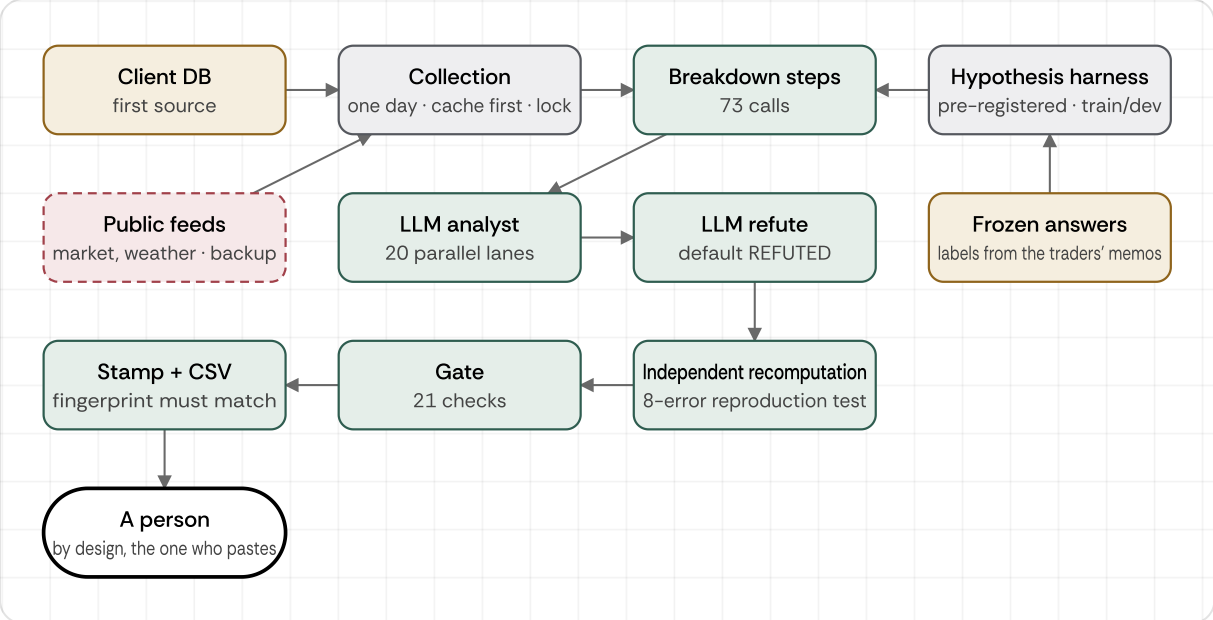
Gates

Pre-registered; train/dev; holdout once
21 gate checks and a publish stamp
Two-way step-table check
No PRs and no human review on record

System architecture

System architecture

Two inputs merge into one collection step and flow in a single line from front to back, and the hypothesis harness, measured against frozen answers, wires only the changes that pass into that line.



Client DB

When the client DB has the same data, it is the first source. Before switching, I cross-checked the data value by value.

Collection

Collects one day, uses the cache first and takes a lock. Outage history never trusts an old cache and is fetched fresh. The analysis starts only once the required inputs are in.

Breakdown steps

Runs fact settling, sensitivity, outage matching and grading (four-axis counterfactual), load, weather, ramp, forecast error and renewables, then flow, neighbouring corridors and trigger notes, in 61 step calls plus 12 soft calls that tolerate partial failure. Stops at the first failure.

Hypothesis harness

Pre-registers a hypothesis, scores it on train/dev with changes to at most three files, then judges adopt, reject or open. Only adopted changes are wired into the outage-grade, condition and lead-trigger steps. The holdout is opened once, when merging to main.

Public feeds

Backup input for the days the DB does not cover: public market feeds and public weather, renewables and generation data.

LLM analyst

The draft-review step. It runs 20 lanes in parallel, and with the next step it takes about 85% of the time.

LLM refute

Runs in refute mode: the default verdict is REFUTED and CONFIRMED needs a cited source. A numeric sentence without a source cannot enter the claim ledger.

Frozen answers

Labels taken from the traders' past hand-written memos and then frozen. The default run does not read these memos (blind).

Stamp + CSV

The publish function writes no file unless the stamp and the fingerprint match. One constraint is one CSV row.

Gate

The gate has 21 numbered checks. A pass writes a stamp that holds the note fingerprint; a failure deletes it.

Independent recomputation

Recomputes separately from the pipeline. It counts as passing only if it reproduces 8 past errors.

A person

By design a person copies and pastes the memo. Nothing is written into the system automatically, and only a person signs in.

`tools/batch/outage_class.py` · Production and scoring call the same grading function and only the input differs. The docstring lists the checks the old backtest skipped. (Docstring condensed)

```
def grade(t, bind_days, window_days, link_rank, stamps, siblings,
          bind_he, sib_hours=None):
    """Final grade for one outage candidate: (kind, why, confidence).

    Production (`classify`) and scoring (`backtest.score`) both call
    this same function (2026-09-14). The earlier backtest chained its
    own gates in its own order. It skipped the matched control group,
    the planned-outage cap, the rotation probe, the abstain for history
    that was never collected, and the link-rank ceiling.

    What differs between call sites is only the input: the observation
    days, that day's 5-minute stamps, the sibling bind hours."""
    own = _own(bind_days, window_days, t, link_rank, stamps,
               siblings, bind_he, sib_hours=sib_hours)
    kind, why, conf = onset_gate(own["kind"], own["why"],
                                 own["confidence"], ...)
    ...
```

`tools/batch/scorecard.py` · A right grade is +1, an abstain 0 and a wrong grade -WRONG_COST. The wrong-answer cost sets the accuracy bar for a new verdict. (Comment condensed)

```
# The cost of one wrong grade (a correct grade = 1). A new grade only
# earns points once its accuracy clears 1/2.
# The basis for 1 is the 09-02 decision: onset alone was passed over
# for onset-or-necessity (52% precision). Raising it to 2 would push
# the new-grade threshold to 2/3.
WRONG_COST = 1.0

def value(ours, memo, w=None):
    """Row score: correct grade +1, abstain 0, wrong grade -w."""
    if ours not in GRADES:
        return 0.0
    return 1.0 if ours == memo else -(WRONG_COST if w is None else w)
```

Key decisions

Four of the six decisions I confirmed or gave as input. The biggest, scoring with the same function production uses, was the agent's move; rolling back hypothesis 04 as a result was my call.

The unit of adoption: a 60/20/20 split by constraint-id hash, connected components of flipped rows My sign-off

Problem

To judge whether a hypothesis really improved things, I first had to decide what counts as one unit and how to split the data. The sample was small, so the statistics wobbled easily.

Options considered

A 4–10-case pilot window as the unit (v1): drawing 1,000 subsets of 8 cases put the quartiles of critical precision at 0.33, 0.50 and 0.75, a 42-point spread, so it was unfit

Split by connected components of the whole corpus (v6): one giant component swallowed 33–71% of the pairs, so it was dropped

✓ A 60/20/20 split by constraint-id hash; the unit is a connected component of flipped rows; dev only confirms direction

Decision

I split train, dev and holdout 60/20/20 by constraint-id hash. The unit of adoption is a connected component of flipped rows grouped by the same constraint, ticket and equipment. Dev only confirms direction, and a power model sets the minimum effect (11 rows on train, later 14).

Result

The first power model treated rows as independent, which inflated the power to 96%; I fixed it with a per-ticket model. This frame was used to judge the 22 hypotheses.

Cost & remaining risk

Dev only confirms direction, so the evidence that a change generalises is weak. Adopting hypothesis 04 rested on 4 dev components, p 0.25. Each cycle runs two backtests of about 79 seconds.

Scoring rule: right +1, abstain 0, wrong -WRONG_COST My sign-off

Problem

To tell whether a hypothesis that issues more verdicts is a good one, the cost of a wrong verdict had to be in the score.

Options considered

Two cells, right or not right: turning an abstain into a wrong answer loses nothing, so guessing is free

✓ Right +1, abstain or unscored 0, wrong -WRONG_COST (WRONG_COST = 1)

Decision

Per row: right +1, abstain or unscored 0, wrong -WRONG_COST. The wrong-answer cost sets the accuracy bar for a new verdict (WRONG_COST ÷ (1 + WRONG_COST)), and turning a verdict back into an abstain earns the same. On September 2 I chose the side that issues more verdicts at 52% accuracy and set the cost to 1.

Result

Under the earlier rule, turning 72 abstains in train into supporting would have been adopted 45 to 0 while publishing 27 wrong answers. After the rule changed, that example fell to open (+18 against a minimum effect of 27 points). Hypotheses 03 to 05 got the same verdict at a wrong-answer cost of 1 and 2.

Cost & remaining risk

Whether to raise the wrong-answer cost to 2 is still open. Raising it would need 67% accuracy for a new verdict.

Score with the same function the pipeline uses

Agent default · no sign-off on record

Problem

Hypothesis 04 had been adopted and merged into main, but the path the backtest scored was not the production path.

Options considered

The backtest chains candidate ranking, flow gate and onset gate in its own order, and misses the matched control group, the planned-outage cap, the neighbouring-corridor test, the abstain for uncollected history and the link ceiling that production applies (the old way)

✓ The backtest and production call the same grading function; only the input differs

Decision

I merged the outage-grade logic into one function that the backtest and production both call. Only the input differs (observation days, that day's 5-minute stamps and so on). Every score since then is taken on the production path.

Result

Measured on the same basis, the agreement fell from +4 points (51%) to -39 points (35%, 5 of 10 critical right). Hypothesis 04, adopted earlier, moved only 1 train row and 1 dev row on the production path, both for the worse, so I rejected it and rolled it back.

Cost & remaining risk

That the work starts at -39 points is a weakness of this model.

Outage grade is physical necessity; advance notice is a separate attribute My sign-off

Problem

The September 2 rule did not let a pre-announced outage be called critical. One cell folded two questions together: "would it have bound without this outage" and "did the day-ahead market know".

Options considered

The September 2 rule: a pre-announced outage cannot be critical (only a surprise to the day-ahead market is critical)

✓ The grade is physical necessity only; whether it was announced goes at the end of the note line as a separate fact

Decision

The grade is decided only by "would it have bound without this outage", and whether it was announced is written separately at the end of the note line.

Result

The split raised the memo agreement by 22 points on train and 13 on dev, and critical precision rose from 55% to 64%.

Cost & remaining risk

As a definition decision it was applied at p 0.068, short of the 0.05 bar for adopting a hypothesis. On the production path critical recall only goes from 2% to 9% (27 of 303).

Do not read the traders' memos by default (blind) My input

Problem

The reference answers are the traders' memos, and if the analysis reads the same memos the comparison becomes circular.

Options considered

Read the memos as input and switch them off with an option (the old default)

✓ The default run never reads the memos; an option switches them on

Decision

I changed the default on August 24. The old commands keep working.

Result

Two leaks of the blind were measured. With only an environment variable set, two tools still read the memos (measured on September 5), and the same sentences sat inside an input dump. I named three doors as forbidden (the memo cache, the author block of the input dump, external viewing) and passed the switch only as an argument flag.

Cost & remaining risk

The generation-outage axis always abstains and the outage grade stands on the four-axis counterfactual alone, so a note with a dead axis becomes normal operation. Why a note got shorter has to be left in a sidecar.

Publishing requires the gate stamp Agent default · no sign-off on record

Problem

On August 17 two CSVs that had not passed the gate were published. The steps had been listed and run by hand, and 14 of 20 working directories had no step log.

Options considered

A person watches each step's exit code (a rule that existed only on paper)

✓ A pass writes a stamp that holds the note fingerprint, and the publish function checks the stamp and the fingerprint

Decision

When the gate passes it writes a stamp with the note fingerprint, and a failure deletes it. The publish function writes no file unless the stamp and the fingerprint match. Four tests cover it: no stamp, normal, note changed after approval, and a failed pass.

Result

I added the publish stamp on August 18, and on the 19th a single entry point and a two-way step-table check. That check found 4 steps that had been missing.

Cost & remaining risk

The stamp guarantees only that "it passed". The items still marked "person" in the rules document are the next work list.

Quality and verification

I rejected five hypotheses that won on train and lost on dev, and the limits of this work are written out below.

I measured quality with the hypothesis harness. The 22 hypotheses came out as 6 adopted, 6 rejected, 8 open and 2 adopted as input (counted by hand from the 22 rows of the experiment table). Every number on this page is a value left in an experiment report or its result JSON; I did not re-run them for this write-up.

What this method caught was false improvements. Five hypotheses were positive on train and negative on dev, so I rejected them; judged on train scores alone they might have been adopted. Some gains held on both, such as experiment 45 (wind-forecast variables from a neighbouring region). The lead-trigger hit rate rose from 22% to 32% on train and from 29% to 46% on dev, with a net gain of +57 rows on train and +27 on dev. Rows where the wind condition was on had 103 hits against 51 misses (67%), and the other conditions 36%.
There are limits I will not hide. Experiment 49 (hold off when a connected outage is mostly scheduled over the next 90 days) was adopted at +59 rows on train, but its dev net gain is 0, and the report itself says the effect shows only on train. The three experiments that switched outage inputs

to the client DB did not raise the net gain (rows), and on the production path critical recall for outages is 9%. The agreement started at -39 points. Eight hypotheses are still open.

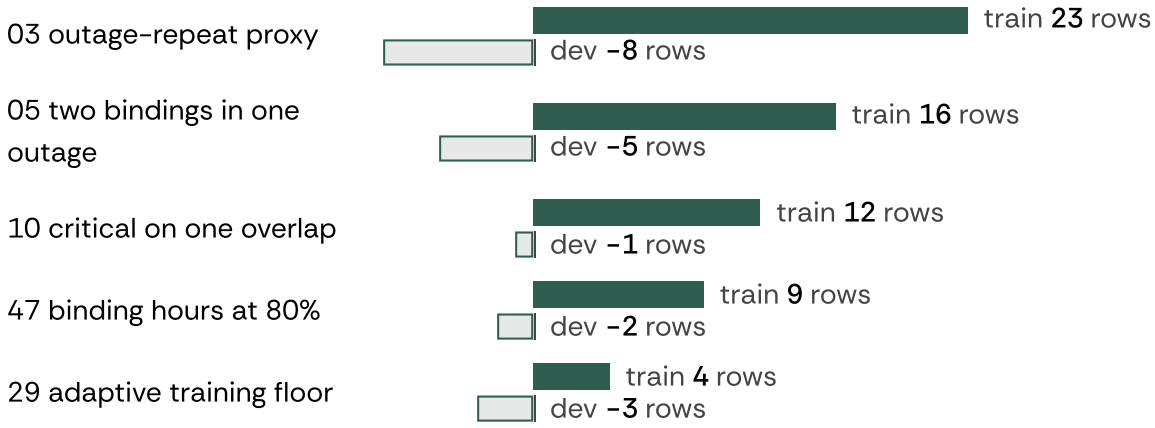
Checks were built up item by item: 19 input-contract checks, the rule "a tool's silence is not the data's silence", and an independent recomputation that passes only if it reproduces 8 past errors. A numeric sentence without a source cannot enter the claim ledger, and what is not in the ledger cannot reach a report. The step table and the script are checked against each other in both directions.

On tests: 110 of the 123 Python tools carry a self-test string. I made no run record this time, so I give no pass count. One 4–10-case window takes about 20 minutes, and a 69-case window takes more than an hour in the LLM steps alone. Running the same window five times, four runs were cut off by a limit or by the gate.

Five hypotheses that won on train and lost on dev

Five hypotheses had a positive net gain on train but a negative one on dev and were rejected.

Source: Net gain = rows fixed - rows broken (the cycle summary table of the experiment registry, 22 rows). Every net gain on this page uses this unit.



The life of one hypothesis

A hypothesis is registered first, scored on train and dev and given a verdict before it is wired into the pipeline, and the same hypothesis can never be registered twice.

1. **Pre-registration** ○Person
   Hypothesis and verdict criteria written as JSON first
2. **Change** ◇Agent
   Three files at most
3. **Score on train and dev** □System
   Split 60/20/20 by constraint-id hash; two backtests of about 79 seconds per cycle

④ **Verdict** System
Adopt, reject or open; the unit is a connected component of flipped rows, and dev only confirms direction

⑤ **Wire in** System
Only adopted changes

⑥ **Holdout once** Person
Opened once, before merging to main

Loops back
↺ ④ Verdict → ① Pre-registration: next hypothesis (the same one cannot be registered twice)

One feature, start to finish

Hypothesis 04 was adopted, merged and rolled back within a day.

This follows one hypothesis from registration to roll-back in a day. The question of the trace is: when did I learn that an approved improvement had changed nothing in production?

Hypothesis 04, from adoption to roll-back

The approval on the old path had no effect on the pipeline output: only 3 of 50 scored pairs existed in the window verdicts, and once scoring matched production the agreement fell from +4 to −39 points and hypothesis 04 was rolled back.

+4 → −39 pts	3 (of 50 pairs)
agreement, old path to production path	scored pairs present in window verdicts

1–7



adoption to roll-back · 1 day

1	Sep 13, 2026	**Hypothesis 04 registered** Raise the "20x density" bar so a single overlap with low day-ahead density is not called critical
2	Sep 13, 2026	**Scored on the old path, adopted** Train net +15 rows (27 to 12), components 13 to 2, p 0.002, dev 6 to 2
3	Sep 14, 2026	**Holdout opened once, merged** 9 fixed, 6 broken, p 0.34: the same direction, p above the 0.05 bar
4	Sep 14, 2026	**Path mismatch found** Only 3 of 50 pairs exist in the window verdicts

5 Sep 14, 2026 **Scored with the production function**
Scoring becomes +1, 0, -1; agreement +4 to -39 points

6 Sep 14, 2026 **Re-measured on the production path, rejected and rolled back**
Only 1 train row and 1 dev row moved, both for the worse

7 Sep 14, 2026 **Outage grade redefined**
Grade is physical necessity, advance notice an attribute; agreement up 22 points on train, 13 on dev

1 Hypothesis 04 registered (Sep 13)

The hypothesis raises the "20x density" bar so that a single overlap with low day-ahead density is not called critical.

2 Scored on the old path, adopted (Sep 13)

Train net +15 rows (27 to 12), components 13 to 2, p 0.002, dev 6 to 2. Adopted.

3 Holdout opened once, merged (Sep 14)

Before merging to main I opened the holdout once. 9 fixed, 6 broken, p 0.34: the same direction but not significant, and I merged it as it stood.

4 Path mismatch found (Sep 14)

Of the 50 pairs the backtest scored, only 3 existed in the production window verdicts. The scored path and the production path differed.

5 Scored with the production function (Sep 14)

I changed the scoring rule to right +1, abstain 0, wrong -1 and scored with the same grading function as production; the agreement fell from +4 to -39 points.

6 Re-measured on the production path, rejected and rolled back (Sep 14)

Measuring hypothesis 04 again on the production path moved only 1 train row and 1 dev row, both for the worse. I rejected it and rolled it back.

7 Outage grade redefined (Sep 14)

I split the grade (physical necessity) from the advance-notice attribute. The agreement rose by 22 points on train and 13 on dev.

Retrospective

The approval on the old path had no effect on the pipeline output, and that only came out once the scorer matched production.

An approved improvement had no effect on the pipeline output. I merged hypothesis 04 although its holdout was not significant (p 0.34), and that the backtest scored a different path than production only came out once the scorer matched production. Since then every score is taken on the production path.

On August 17 I listed and ran the steps by hand, and two CSVs that had not passed the gate were published. The rule was in a document but nothing checked it. I replaced it with the publish stamp and a two-way step-table check, and split the skill document, which had grown to 1,523 lines, into run order, rules with checks, and an incident log.

Things have passed while hollow. All 233 binding days were "no supporting cause" and the check still passed, the linter failed to load and reported "0 violations", and a travel-leg feed missing 5 days kept the same file name and flipped a threshold verdict. That led to the input-contract checks and the rule that a tool's silence is not the data's silence.

Explanations I wrote without checking were wrong twice in a day. I then set a rule to write down the whole state before answering. Running a big window (69 cases) repeatedly and being cut off by a limit led to the rule of fixing on small windows first.

Some things are still open: 8 open hypotheses, the decision on raising the wrong-answer cost to 2, critical recall for outages that stays at 9% on the production path, and experiment 49 with a dev net gain of 0.

How I work

This project was also built with a role-based AI agent team. The team and its gates differ from project to project; this one's are under "My role" above. Below is the method as written up while building the EWA platform.

See how I set up and ran the work on the 2026 EWA platform →

https://jake-jo-portfolio.vercel.app/en/process/

Contact

whxoans@gmail.com
+82 10-9233-7440