Research Note 001
From Framework to Pipeline: Implementing the Stage-Gating Loop

Telemetry → Veto Discovery → Promotion (evidence-backed)

Sean Plows
Lucitech Computer Solutions

27 February 2026

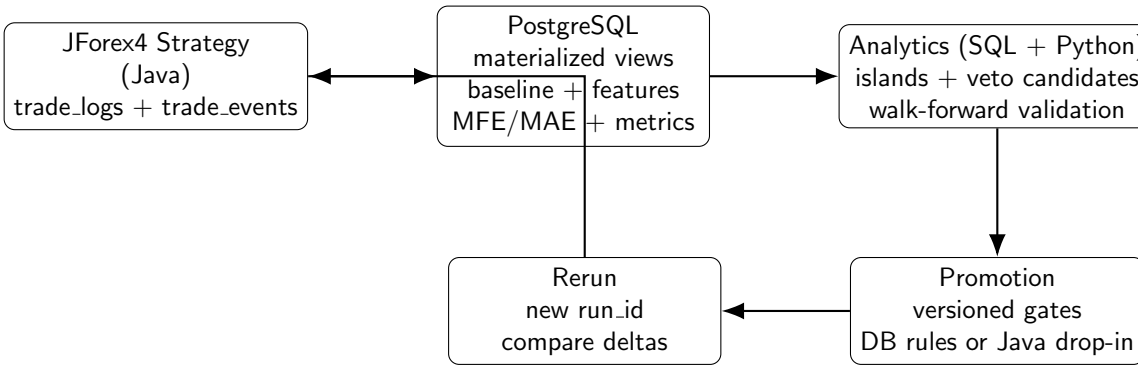
How this follows the White Paper (Part 1)

Part 1 defines the research programme and notation: a two-stage funnel.

1) **Stage gating (this note):** discover interpretable veto rules that remove low-quality regions of feature space and improve conditional expectancy out-of-sample.
2) **Stage 2 (later):** microstructure entry refinement using tick-level variables, only after Stage 1 is stable.

This note provides the missing bridge: **framework** $\rightarrow$ **working pipeline**, with concrete artefacts for each claim.

The Stage-Gating Loop (Telemetry → Veto Discovery → Promotion)



Key property: every iteration is tagged (run_id, pass number, simulation method) and produces bundled artefacts.

Evidence map (what proves what)

Each Part 1 claim is backed by a concrete artefact:

Telemetry + lifecycle logging: JForex strategy (Java) writes trade lifecycle + events (DB).

Deterministic DB build: `run_pipeline.sh` refreshes/exports the generic-first view stack.

Cluster discovery ("islands"): `db_island_finder_dropin.py` segments pockets and ranks good/bad regions.

Veto discovery + validation: `veto_validation_harness_db_v3` produces screened candidates + summary.

Promotion back to runtime: vetted rules are translated into gates (DB-driven rules or Java drop-in).

The goal is reproducibility: same input run_id $\Rightarrow$ same exports and reports (fail-fast if counts are zero).

Implementation stack & provenance (engineering receipts)

This project is deliberately end-to-end: research method + production-grade execution loop.

Strategy runtime: Java strategy under Dukascopy JForex4; deterministic config/CLI; emits trade lifecycle + event telemetry.

Data plane: PostgreSQL (`algo`/`model`/`market`) with materialized views for baseline, features, MFE/MAE, and trade metrics.

Analytics harness: SQL refresh/export + Python validation (walk-forward, bootstrap, multiple-testing control) producing versioned report bundles per `run_id`.

Ops & repeatability: Docker/Compose per instrument; health checks + restart policy; pinned JForex version; writable cache mount for deterministic runs.

Provenance & CI/CD: GitLab CI builds versioned JAR artefacts; deployment is "artifact $\rightarrow$ symlink $\rightarrow$ service recreate". Each run links `run_id` $\leftrightarrow$ git SHA $\leftrightarrow$ parameter hash $\leftrightarrow$ dataset window.

Sensitive details (credentials, exact thresholds, private infra, deploy keys) are excluded; the goal is auditability and reproducibility.

Implementation: generic-first DB pipeline (materialized views)

Generic-first allows one analytics harness across instruments, with optional pair-specific overrides.

```
# run_pipeline.sh (conceptual)
# baseline -> features -> mfe/mae -> metrics
BASELINE_MV = baseline_trades_v3 (or v3_<ccy>)
FEATURES_MV = features_at_fill_generic
MFE_MV      = mfe_mae_ticks_generic
METRICS_MV  = trade_metrics_trades_v6_generic (fallback v5)

deps refreshed in order; exports bundled per run_id
```

This is the “plumbing” that turns Part 1 into a repeatable experiment loop.

Mathematical frame: high-dimensional trade space

Each trade is represented at decision time by a feature vector:

$$\mathbf{x}_i \in \mathbb{R}^d$$

Outcome target examples:

$$y_i \in \{\text{profit}_i,\ \text{profit}_i/\text{risk}_i,\ \text{profit}_i/\text{ATR}_i\}$$

We search for subsets ("pockets") defined by a bucketing function $b(\mathbf{x})$:

$$b(\mathbf{x}) = (\text{dir}, \text{hour}, \text{ATR-quantile}, \text{spread-bin}, \ldots)$$

Veto is a conservative operator: forbid a pocket only if it improves outcomes *and* survives walk-forward validation.

MFE/MAE: excursions as post-trade evidence (Stage 2 boundary)

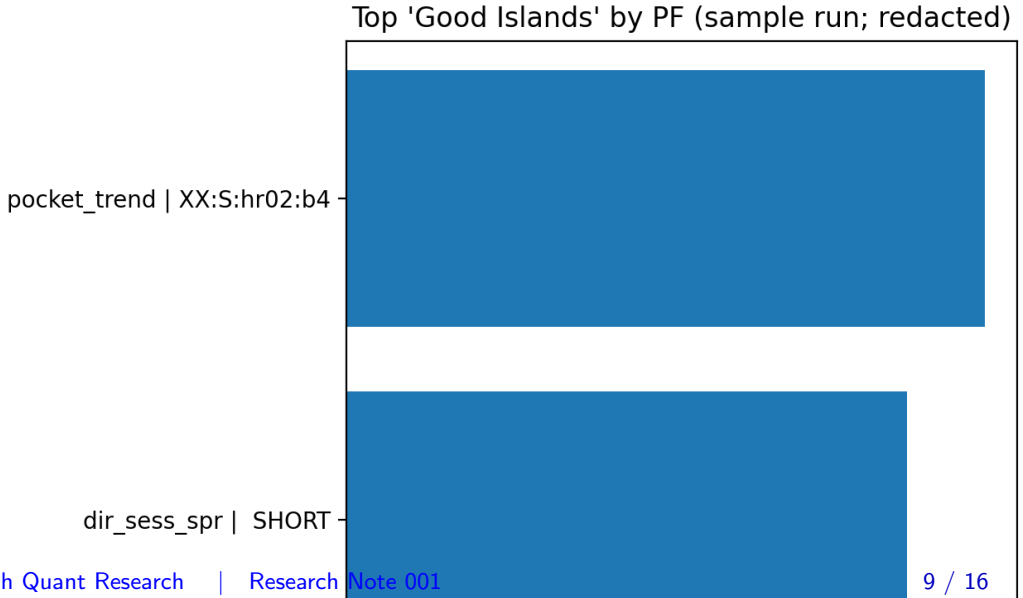
Given entry time t_i and directional move $\Delta P_i(t) = d_i(P_t - p_i)$ over horizon H:

$$\mathrm{MFE}_i = \max_{t \in [t_i, t_i+H]} \Delta P_i(t), \qquad \mathrm{MAE}_i = \min_{t \in [t_i, t_i+H]} \Delta P_i(t)$$

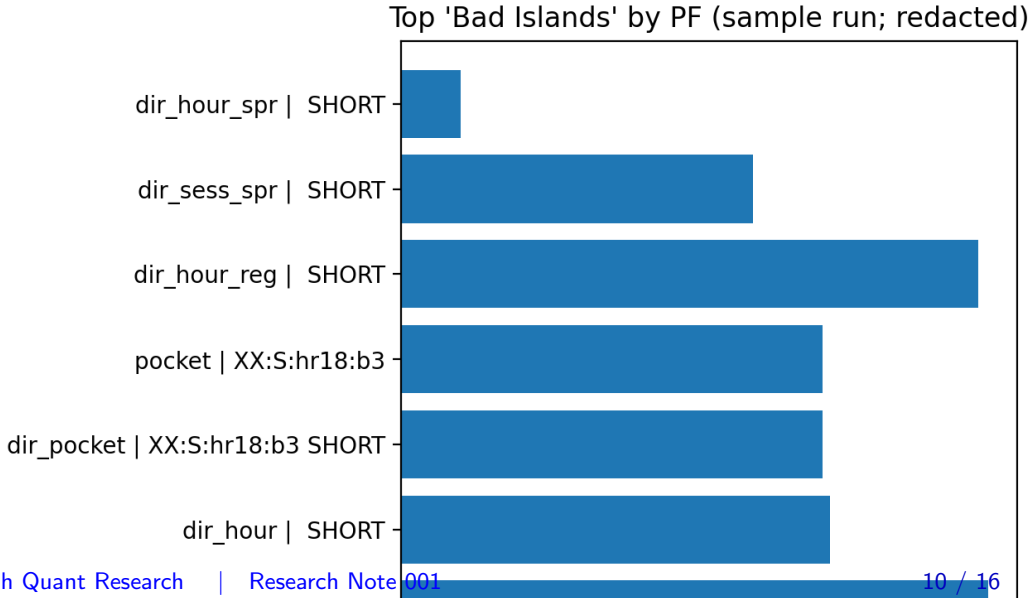
Stage 1 uses decision-time features only. MFE/MAE provides **post-trade evidence** for:

- whether exits are leaving money on the table (MFE)
- whether entries are exposed to adverse excursion (MAE)
- whether Stage 2 (microstructure entry refinement) is justified

Example output: “good islands” (profit factor surface)



Example output: “bad islands” (profit factor surface)



Validation hygiene: walk-forward + bootstrap + multiple testing

The pipeline is designed to resist false discovery:

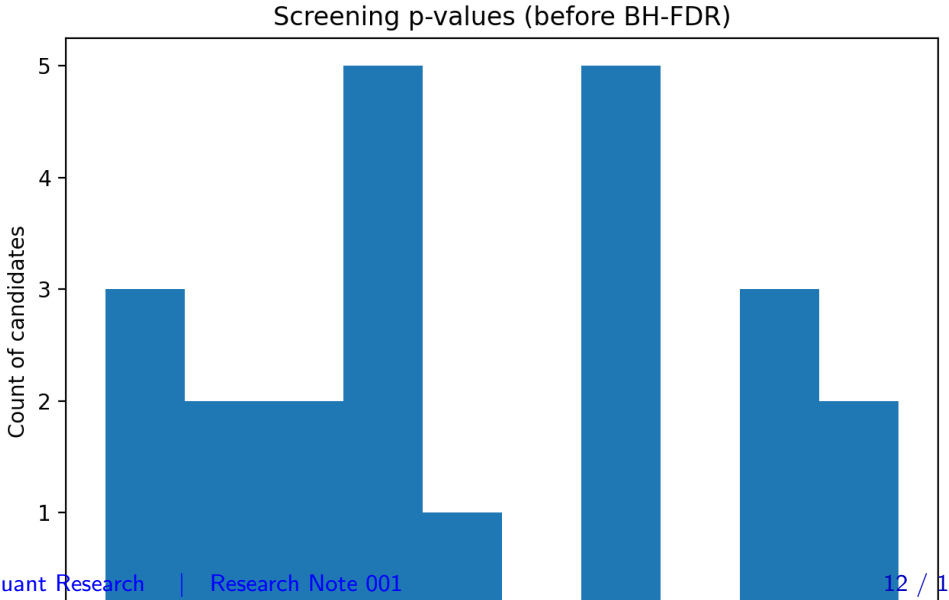
Purged / embargoed folds: avoid temporal leakage in time-series validation.

Block/bootstrap uncertainty: estimate stability of pocket deltas under dependence.

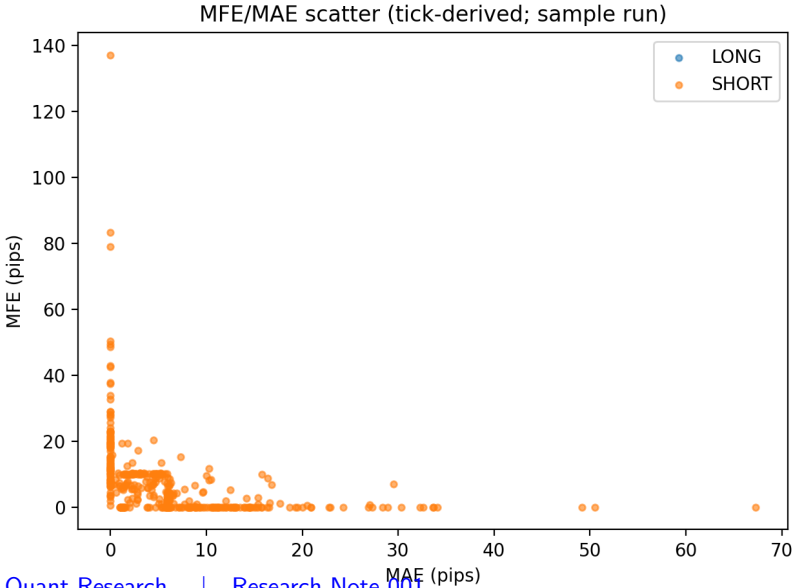
Multiple-testing control: screen p-values and apply BH-FDR; no promotions if everything is rejected.

A veto is promoted only if it is:
(i) statistically supported across folds,
(ii) economically meaningful after costs, and
(iii) conservative (reduces risk of overfitting).

Example output: screened candidate p-values



Example output: MFE vs MAE scatter



This supports the

Promotion: from vetted veto → runtime gate

Promotion is deliberately boring:

- create a versioned rule-set (DB) or code-level gate (Java)
- rerun live/demo and compare against baseline with identical telemetry
- monitor drift (when veto effectiveness decays, revisit)

```
# Conceptual: gate evaluation at decision time
if (veto_rule_applies(features_at_fill)) {
  skip_trade();
}
```

No deployable thresholds are disclosed here; the paper focuses on method + implementation proof.

The timelapse story (next publication step)

The next step is to publish a time-ordered sequence showing the loop in action:

1) start with a raw-ish run (baseline) over a fixed window
2) discover candidate veto pockets
3) validate and promote a conservative subset
4) rerun and measure deltas (trade count vs stability metrics)
5) repeat until a simple, boring live micro-lot run is stable over n months

The emphasis is on **process control** and **reproducible evidence**, not on selling an edge.

Close

Summary:

Part 1 established the methodology and anti-overfit safeguards.

This note shows the working pipeline: telemetry $\rightarrow$ islands $\rightarrow$ veto validation $\rightarrow$ promotion.

Next: timelapse across iterations and n months of stable live demo/micro-lot evidence.

Online hub: https://lucitech.co.uk/lucitech-quant-research/