Knowing When to Stop: Pertura for Graph-Enforced PI Gating in Perturb-seq Agents

Tianyi Wang [*1]

Abstract

LLM-based scientific agents can generate and execute analysis code, but may proceed past unresolved experimental-design assumptions in complex assays such as Perturb-seq. Here we present Pertura, a Perturb-seq analysis agent that represents workflows as stateful phases with declarative preconditions, blocking PI interactions, and durable structured state. Phase-entry gates constrain downstream interpretation until assumptions about perturbation modality, controls, and annotation policy are specified, resolved, or confirmed by a human expert. On the Norman/Weissman Perturb-seq dataset (GSE133344), Pertura performs matrix-level QC, guide/target audit, CRISPRa sanity checks, and direction-of-effect interpretation. A no-gate ablation logs the same failed prechecks but permits phase entry, producing an unresolved boundary-crossing event; this shows that graph enforcement prevents unresolved design-boundary crossings at phase boundaries.

1. Introduction

LLM-based agents increasingly combine reasoning, tool use, and code execution (Yao et al., 2023; Wang et al., 2024), making them promising interfaces for scientific data analysis. In computational biology, recent agentic systems such as CellVoyager (Alber et al., 2026) and CellAgent (Xiao et al., 2026) have shown that agents can generate de novo analytical insights and execute refined single-cell analysis pipelines. However, the central risk is not merely whether an agent can invoke the correct library call, but whether it should proceed when the experimental design remains underspecified. This risk is especially acute in Perturb-seq, which couples pooled genetic perturbations with single-cell RNA-seq readouts (Dixit et al., 2016). Downstream interpretation depends on design choices such as perturbation modality, non-targeting control definitions, guide-to-target assignment, single- versus multiplexed-perturbation policy, and quality-control thresholds. If these assumptions remain unresolved, an agent may still produce plausible code and biologically fluent summaries while silently crossing a scientific boundary that should have required confirmation from a domain expert.

Prompt-level guardrails provide only weak protection against this failure mode. A language model can be instructed to ask for clarification, but the decision to stop remains inside the model's own generation loop. Reasoning-and-acting frameworks such as ReAct (Yao et al., 2023), and self-reflection methods such as Reflexion (Shinn et al., 2023), can improve agent trajectories, but they do not by themselves prevent execution past unresolved design state. For perturbation analysis, we argue that PI supervision should not be represented merely as conversational advice; it should be represented as an execution constraint.

We present Pertura, a Perturb-seq analysis agent that converts principal investigator (PI) supervision into graph-enforced workflow control. Pertura organizes analysis as a stateful sequence of scientific phases, where each phase is specified by an `AnalysisNodeSpec` declaring preconditions over durable structured state, allowed tools, and expected artifacts. Before entering a phase, the runtime evaluates these preconditions. Missing PI-level assumptions can trigger graph-level prechecks or blocking PI interactions, both of which are recorded in durable structured state before downstream analysis proceeds.

We make the following key contributions.

- We introduce declarative phase preconditions as a first-class abstraction for scientific workflow scope, encoding Perturb-seq SOP requirements as checks over structured analysis state.
- We implement a stateful PI-gating layer that combines phase-entry prechecks, blocking PI interactions, structured state records, and a no-gate ablation mode for testing enforcement.

[1]School of Life Science, Peking University, Beijing, China. Correspondence to: Tianyi Wang <wangtianyi2005@stu.pku.edu.cn>.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026)

- We evaluate Pertura on the Norman/Weissman GSE133344 Perturb-seq dataset (Norman et al., 2019), comparing a gated run against a no-gate ablation that disables phase-entry enforcement.

2. Method

2.1. Workflow Formalization

Pertura formulates perturbation analysis as a stateful graph execution problem, following the stateful workflow abstraction used in LangGraph-style agent runtimes (LangChain, 2025). Let $\mathcal{S}_t \in \mathcal{S}$ denote the durable structured analysis state at step t. A workflow is specified by an ordered standard operating procedure (SOP) $\Pi = (v_1, \ldots, v_K)$, where each phase v_k is represented by an `AnalysisNodeSpec`:

$$v_k = (\mathrm{id}_k, \mathrm{purpose}_k, \mathcal{P}_k, \mathcal{T}_k, \mathcal{R}_k, \mathcal{W}_k, \mathcal{C}_k, \mathcal{A}_k).$$

Here $\mathcal{P}_k$ is a set of declarative preconditions, $\mathcal{T}_k$ is the tool set exposed to the phase, $\mathcal{R}_k$ and $\mathcal{W}_k$ describe expected state reads and writes, $\mathcal{C}_k$ denotes completion criteria, and $\mathcal{A}_k$ specifies expected artifacts. Thus, a scientific phase is not only a natural-language instruction, but a structured execution unit whose admissibility is defined with respect to the current state.

We represent each precondition as a predicate $p_{k,j} : \mathcal{S} \to \{0, 1\}$. The set of unmet preconditions for phase v_k at state $\mathcal{S}_t$ is

$$M_k(\mathcal{S}_t) = \{p_{k,j} \in \mathcal{P}_k : p_{k,j}(\mathcal{S}_t) = 0\}.$$

In the Norman/Weissman case study, the SOP includes data inspection, experimental design confirmation, matrix-level QC, guide/target audit, perturbation sanity checking, target-level QC, and report generation.

2.2. Graph-Enforced PI Gating

At each phase boundary, Pertura applies a graph-level gate before allowing execution:

$$G_k(\mathcal{S}_t) = \begin{cases} \mathrm{enter}(v_k), & \text{if } M_k(\mathcal{S}_t) = \emptyset, \\ \mathrm{block}(M_k(\mathcal{S}_t)), & \text{otherwise.} \end{cases}$$

If the gate returns ENTER, the phase body executes. If the gate returns BLOCK, the graph suspends execution and returns control to the PI. The PI response is converted into a structured state patch δ_t, which is applied as

$$\mathcal{S}_{t+1} = \mathrm{Apply}(\mathcal{S}_t, \delta_t).$$

The same gate is then re-evaluated on $\mathcal{S}_{t+1}$; phase execution resumes only if $M_k(\mathcal{S}_{t+1}) = \emptyset$.

This design places the decision to halt or proceed outside the model generation loop. The language model may propose analyses, summarize results, or request PI input, but it cannot enter a gated phase while the graph runtime observes unsatisfied phase-entry preconditions. PI supervision is therefore implemented as workflow control rather than as prompt-level advice.

In perturbation analysis, gated fields include perturbation modality, control definition, guide column, target column, and policy for multi-guide or dual-target cells. These fields determine the interpretation of downstream perturbation effects, so Pertura treats them as phase-entry requirements rather than optional conversational context.

The enforcement scope is phase-level. Within a phase, the agent may invoke tools, execute Python code, and request voluntary PI input. Pertura does not claim to police every intermediate reasoning or coding action. Instead, it enforces structured preconditions at phase boundaries, where unresolved experimental-design assumptions become consequential for downstream validity.

2.3. Jupyter-Grounded Execution and Audit Trail

After a phase passes its preconditions, Pertura executes analysis actions in a persistent Jupyter kernel. Each tool invocation is recorded with its input, generated code when applicable, stdout and stderr streams, execution status, errors, and produced artifacts. The run directory stores the executed notebook, event logs, tool-call traces, structured state snapshots, generated CSV tables, figures, and the final report.

This execution layer makes graph-level decisions inspectable. For any completed run, a user can recover which phase was active, which preconditions were evaluated, which fields triggered a blocking event, what PI response or scripted demo policy updated the state, and which tools executed only after the gate passed. Thus, the audit trail links phase-entry control decisions to concrete computational artifacts.

2.4. No-Gate Ablation

To isolate the effect of phase-entry enforcement, we compare Pertura with a no-gate ablation. Both settings use the same dataset, SOP, agent prompt, and tool set. The only difference is the treatment of failed phase-entry preconditions. In the gated setting, $M_k(\mathcal{S}_t) \neq \emptyset$ triggers a block and blocks phase entry. In the no-gate setting, the graph logs the failed precheck but allows execution to continue.

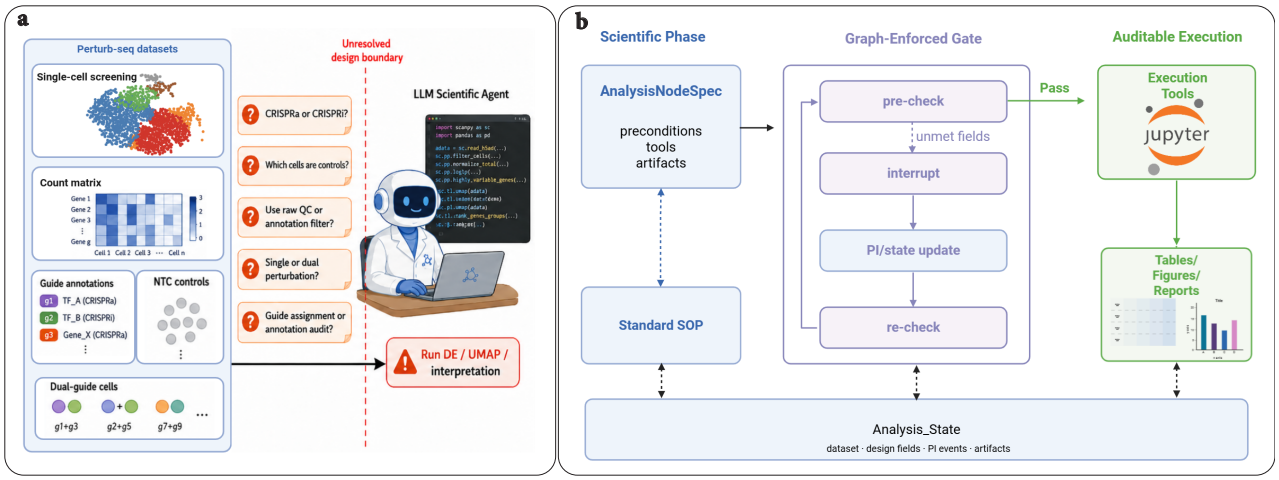

Figure 1. **Motivation and mechanism of Pertura.** (a) Perturb-seq analysis depends on PI-level design assumptions such as perturbation modality, control labels, guide assignment policy, and dual-perturbation handling. Prompt-only agents may proceed with plausible code even when these assumptions remain unresolved. (b) Pertura converts these assumptions into stateful workflow constraints. The runtime evaluates each active `AnalysisNodeSpec` against durable structured state; unresolved fields can trigger phase-entry prechecks or blocking PI interactions, and no-gate ablation disables the phase-entry enforcement path.

3. Evaluation

3.1. Implementation Details

All reported gated and no-gate ablation runs use the same code snapshot, prompt template, tool set, LLM backend, and decoding configuration. The backend is a single hosted model accessed through the same API-compatible interface in both settings. The only intervention is the phase-entry enforcement flag; all ablation metrics are computed from persisted phase-boundary events.

3.2. Dataset and Scope

We evaluate Pertura on Norman/Weissman GSE133344, a canonical CRISPRa Perturb-seq screen with public raw 10x-style matrix files and curated cell-identity annotations (Norman et al., 2019). We choose this dataset because it contains design choices typical of Perturb-seq analysis, including perturbation modality, control definition, guide-label parsing, and dual-guide interpretation, while providing enough public annotation to audit the agent's decisions.

Our evaluation has two goals. First, we test whether a full Norman workflow can record and use PI(or user's) guidance while producing a coherent QC and CRISPRa sanity summary. Second, we isolate the effect of phase-entry enforcement using a targeted no-gate ablation. We do not claim FASTQ-level preprocessing, de novo guide assignment, or biological discovery benchmarking.

3.3. Gated Norman Run

In the full gated Norman run, Pertura processes the raw matrix and annotation table under the SOP described in Section 2. The run records blocking PI interactions when design decisions are unresolved. Table 1 summarizes these interactions.The most consequential correction concerned perturbation modality: downstream effect interpretation required resolving the assay as CRISPRa before the agent could proceed.Additional PI guidance resolved the primary cell-calling policy, annotation usage, control definition, and the disposition of unannotated matrix-level cells.

Table 1. **Blocking PI interactions in the Norman run.** Each interaction records a design decision that constrained subsequent analysis.

Phase	Missing field(s)	Applied PI guidance
Inspection	Perturbation modality; guide metadata policy	Confirmed CRISPRa activation; treated `guide_identity` as pre-assigned annotation
Inspection	Cell-calling policy; annotation usage	Used matrix-level QC as primary; kept `good_coverage` and `cellranger_called` audit-only
Design	Control definition; unannotated cells	Used pure NTC controls; retained unannotated cells for QC audit only

3.4. Norman Analysis Summary

After these decisions were resolved, Pertura performed matrix-level cell calling, audited annotation coverage, separated QC-only unannotated cells from the perturbation-analysis population, and ran a CRISPRa direction-of-effect sanity check using standard AnnData/Scanpy-style single-cell data structures (Wolf et al., 2018).

Table 2 summarizes the main results used in the paper. The workflow called 126,315 cells from the matrix-level QC

Table 2. **Norman/Weissman gated-run summary.** The workflow separates matrix-level cell calling from annotation-based perturbation analysis.

Category	Metric	Result
Raw input	Matrix size	5.90M barcodes $\times$ 33,694 genes
Cell calling	Matrix-level called cells	126,315
Annotation audit	Annotated cells	114,870 with `guide_identity`
Annotation audit	Unannotated cells	11,445 QC-audit only
Controls	Pure NTC controls	12,015 cells
Perturbations	Targeting cells	102,855 across 106 target genes
Sanity check	CRISPRa direction	91/102 targets up-regulated (89.2%)
Sanity check	Mean target effect	mean $\log_2$FC $= +0.79$

rule, of which 114,870 had curated guide annotations. The remaining 11,445 cells were treated as unannotated audit-only cells. Among the annotated cells, 12,015 pure NTC cells were used as controls, and 102,855 cells carried at least one targeting guide across 106 target genes. The CRISPRa sanity check found that 91 of 102 tested targets were up-regulated relative to pure NTC controls, consistent with the expected direction of effect for an activation screen. The coverage audit reports 106 annotated target genes, while the direction-of-effect sanity check is computed for the 102 targets with a valid target-gene expression readout in the AnnData gene index.

We interpret these results as a sanity check of the resolved experimental design, not as a claim of new biological discovery. The key point is that downstream direction-of-effect interpretation was performed only after perturbation modality, control definition, annotation usage, and unannotated-cell handling were explicitly resolved.

3.5. No-Gate Ablation

The full Norman run demonstrates that Pertura can execute a PI-gated Perturb-seq workflow, but it does not by itself isolate the causal effect of the graph gate: earlier phases may resolve design fields before the perturbation-effect phase is reached. We therefore run a targeted phase-boundary ablation. Both settings use the same code snapshot, case manifest, target phase (`perturb_effect`), prompt, and tool set. The manifest provides the Perturb-seq dataset path but intentionally leaves PI-level design fields unresolved, including perturbation modality, guide/target columns, control labels, and analysis-population policy.

Table 3 reports mechanism-level behavior rather than downstream biological accuracy. In both settings, the phase-entry precheck detects the same three unmet PI-level preconditions. In the gated setting, execution stops at a pending PI gate and no unresolved phase boundary is crossed. In the no-gate setting, the same failed precheck is logged but bypassed, producing one unresolved boundary crossing. This result supports our central claim: graph enforcement changes whether an agent can proceed past unresolved scientific design state.

Table 3. **Targeted no-gate ablation at perturbation-effect phase entry.** Both runs use the same code snapshot, manifest, target phase, prompt, and tool set; only the precheck-gate ablation flag differs.

Metric	Gated	No-gate
Target phase	`perturb_effect`	
Phase-entry prechecks evaluated	1	1
Failed PI-level prechecks	1	1
Unmet PI-level preconditions	3	3
Pending PI gate recorded	1	0
Unresolved boundary crossings	0	1
Bypass events logged	0	1

This ablation does not require the no-gate run to yield a wrong biological answer. Its purpose is to isolate the operational effect of the gate: with enforcement, unresolved state blocks or constrains phase entry; without enforcement, the same agent can cross the boundary and continue.

4. Discussion

Pertura targets a narrow but consequential failure mode in scientific agents: an LLM may continue a biological analysis even when essential experimental-design assumptions remain unresolved. Rather than proposing a new Perturb-seq discovery pipeline, Pertura aims to frame this problem as a workflow-control mechanism. Perturbation modality, control definition, guide interpretation, and QC policy are represented as structured phase-entry preconditions rather than as informal conversational context which LLM may overlook or mis-interpret.

The Norman/Weissman case study illustrates the practical effect of this design. With gating enabled, unresolved design fields block or constrain phase entry until PI input is converted into structured state. In the no-gate ablation, the same agent, prompt, dataset, SOP, and tools can cross the phase boundary after the failed precheck is logged. This isolates the main effect of Pertura: graph-level enforcement prevents unresolved boundary crossings that prompt-level instructions alone do not forbid.

Several limitations remain. Enforcement applies at phase transitions, not to every generated Python action or tool call, while each action should have its own prechecks and bounded artifacts which may also be termed as constraints. The analysis starts from 10x-style matrix inputs rather than FASTQ-level preprocessing, and guide handling audits curated `guide_identity` annotations rather than assigning guides from raw guide-count matrices. Also, the evaluation is limited to one Perturb-seq dataset and one LLM backend.

Yet, as a prototype, Pertura demonstrates that lightweight runtime gating can make experimental-design ambiguity explicit, testable, and enforceable in scientific-agent workflows.

These limitations suggest several directions for future work. Future versions of Pertura should support more persistent and inspectable runtime sessions, generalize graph-enforced phase gates into evidence-chain gating for biological conclusions, and extend enforcement to finer-grained tool-, action-, and code-level contracts. Together, these directions would move Pertura from a phase-gated prototype toward a broader runtime substrate for accountable Perturb-seq analysis agents.

References

Alber, S., Chen, B., Sun, E., Isakova, A., Wilk, A. J., and Zou, J. Cellvoyager: Ai compbio agent generates new insights by autonomously analyzing biological data. *Nature Methods*, 23(4):749–759, 2026. doi: 10.1038/s41592-026-03029-6.

Dixit, A., Parnas, O., Li, B., et al. Perturb-seq: Dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *Cell*, 167(7):1853–1866.e17, 2016.

LangChain. LangGraph: Low-level orchestration framework for stateful agents, 2025. URL https://docs.langchain.com/oss/python/langgraph/overview. Accessed: 2026-05-08.

Norman, T. M., Horlbeck, M. A., Replogle, J. M., et al. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H., and Ji, H. Executable code actions elicit better llm agents. In *International Conference on Machine Learning*, 2024.

Wolf, F. A., Angerer, P., and Theis, F. J. SCANPY: Large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1):15, 2018. doi: 10.1186/s13059-017-1382-0.

Xiao, Y., Liu, J., Zheng, Y., Jiao, S., Hao, J., Xie, X., Limingzhi, Wang, R., Ni, F., Li, Y., Wang, Z., Shang, X., Bao, Z., Yang, C., and Peng, J. Cellagent: Llm-driven multi-agent framework for natural language-based single-cell analysis. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BsA2GNkJhz.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

A. Reproducibility and Evidence Dossier

This appendix documents the evidence trail used in the main text. The main Norman run supports Tables 1 and 2; the targeted gated/no-gate runs support Table 3. Each run stores an executed notebook, event log, tool-call trace, structured state snapshot, messages, generated artifacts, and a run report. We use these artifacts as a forensic record of what the agent did, which phase boundaries were checked, and which PI-level decisions were recorded.

Table 4. **Run artifacts used for evidence extraction.**

Artifact	Used for	Contents
`analysis_state.json`	PI/state evidence	Datasets, experimental-design fields, PI interactions, artifacts, and SOP state
`case_manifest.yaml`	Ablation setup	Dataset path, target phase, ablation flag, and intentionally unresolved PI-level fields
`events.jsonl`	Gate and ablation evidence	Phase prechecks, pending PI gates, targeted start node, and ablation bypass events
`notebook.ipynb`	Analysis evidence	Executed Python cells, stdout/stderr, and generated analysis outputs
`traces/tool_calls.jsonl`	Tool trace	Tool names, inputs, outputs, and execution status
`run_report.md`	Human-readable report	Dataset summary, PI events, analysis outputs, and final run summary

B. Workflow Contract and Phase Preconditions

Each Pertura phase is represented as an `AnalysisNodeSpec`, which makes the phase contract explicit rather than leaving it only in a prompt. Table 5 summarizes the fields used by the runtime.

Table 5. **`AnalysisNodeSpec` fields.**

Field	Meaning
`node_id`	Unique phase identifier used by the graph runtime
`purpose`	Scientific objective of the phase
`preconditions`	Predicate keys evaluated against structured state before phase entry
`available_tools`	Tool set exposed to the agent within the phase
`reads_state` / `writes_state`	Expected state dependencies and updates
`completion_criteria`	Conditions used to decide whether the phase is complete
`expected_artifacts`	Tables, figures, reports, or summaries expected from the phase

For the targeted ablation, the `perturb_effect` phase used the following phase-entry preconditions:

{`has_dataset_path`, `na_is_perturbation_data`, `needs_pi_design_resolved`, `needs_pi_controls_defined`, `needs_`

The first two conditions establish that the phase is applicable to a loaded Perturb-seq dataset. The last three are PI-level requirements: perturbation interpretation and guide/target fields must be resolved, control labels must be defined, and the analysis-population policy must be specified.

C. PI Interaction Records

Table 6 expands the blocking PI interactions summarized in Table 1. These records are stored in `analysis_state.json` . We report them as design constraints on later analysis rather than as biological findings.

Table 6. **Blocking PI interactions recorded in the Norman run.**

Interaction	Node	Question type	PI answer summary
`pi_20260508_105017`	`data_inspection`	Perturbation modality	CRISPRa activation, not CRISPRi
`pi_20260508_105609`	`data_inspection`	Cell-calling and annotation policy	Matrix-level QC is primary; `good_coverage` and `cellranger_called` are audit-only fields
`pi_20260508_112611`	`experimental_design`	Controls and unannotated cells	Use pure NTC controls; exclude unannotated cells from perturbation comparisons

The full Norman run used blocking PI interactions to resolve design choices before downstream analysis proceeded. Table 7 summarizes how these interactions constrained subsequent execution.

D. Targeted No-Gate Ablation Evidence

The targeted ablation isolates phase-entry enforcement. Both settings use the same code snapshot, case manifest, target phase, prompt, and tool set. The manifest provides the dataset path and assay family, but intentionally leaves PI-level design

Table 7. **Downstream constraint evidence for blocking PI interactions.**

Interaction	Design constraint resolved	Downstream phase constrained	Result
pi_20260508_105017	Perturbation modality	Effect interpretation	CRISPRa direction used before sanity check
pi_20260508_105609	Cell-calling and annotation policy	Matrix-level QC / annotation audit	Annotation fields treated as audit-only
pi_20260508_112611	Controls and unannotated-cell policy	Perturbation comparisons	Pure NTC controls used; unannotated cells excluded

fields unresolved. Table 8 summarizes the persisted event-log evidence used to construct Table 3.

Table 8. **Persisted event evidence for Table 3.**

Setting	Event	Result	Unmet PI-level preconditions
Gated	phase_precheck	blocked_needs_pi	design, controls, analysis_population
Gated	phase_precheck_pi_gate_pending	Pending PI gate	Same three
No-gate	phase_precheck	blocked_needs_pi	Same three
No-gate	ablation_precheck_bypassed	Bypass logged	Same three

The no-gate run later encountered a provider-side LLM API rejection. This occurred after the ablation_precheck_bypassed event had already been persisted. Since the ablation metric is defined at phase-boundary entry, Table 3 uses only precheck, gate, and bypass events, and does not depend on downstream LLM completion quality.

E. Norman Numeric Evidence Sources

Table 9 lists the notebook outputs used to construct Table 2. These values are extracted from the executed notebook stdout and final run report. The target coverage audit reports 106 annotated target genes. The direction-of-effect sanity check is computed for the 102 targets with a valid target-gene expression readout in the AnnData gene index, yielding the 91/102 numerator used in Table 2.

Table 9. **Sources for Norman summary metrics.**

Metric	Value	Source description
Raw matrix size	33,694 genes $\times$ 5,898,240 barcodes	Matrix parsing output
Nonzero entries	442,237,190	Matrix parsing output
Matrix-called cells	126,315	Detected-gene threshold output
Annotated cells	114,870	Cell-identities overlap output
Unannotated cells	11,445	Matrix-called minus annotated output
Pure NTC controls	12,015	Guide-classification output
Targeting cells	102,855	Target annotation output
Unique target genes	106	Target coverage output
CRISPRa up-regulated targets	91/102 (89.2%)	Broad CRISPRa sanity-check output
Mean target $\log_2$FC	+0.787	Broad CRISPRa sanity-check output