Interactive Benchmarking of Scientific Coding Agents for Spatial Transcriptomics Alignment

Anonymous Authors[1]

Abstract

Building and evaluating scientific coding agents requires not only executable tasks but also human-in-the-loop verification for ambiguous and non-verifiable outcomes. Most benchmarks to date have been built bespoke by experts and scored via rubrics on final outputs. Here we explore two interactive augmentations: (1) building benchmark datasets from peer-reviewed papers, and (2) inspecting agent traces to see how workflows emerge. We construct 40 spatial transcriptomics alignment tasks, a challenging problem in computational biology where agents submit coordinate tables aligning a pair of 2D slices. Across 120 runs under each of three configurations—basic prompt, package-aware prompt, and full prompt plus prebuilt virtual environment—we find that more package hints increased tool exploration but did not improve performance: the full regime scored lower than Basic (0.361 vs. 0.428; 95% CI $[-0.113, -0.028]$). Trace inspection shows that richer scaffolding encouraged unnecessary transformations on already-aligned inputs, fragile package-first workflows, and infrastructure failures. Our results highlight that scientific-agent evaluations should therefore involve interactive construction and trace inspection to balance the variance and bias introduced by added context and tools.

1. Introduction

The advance of artificial intelligence (AI) capabilities is rapidly changing scientific practice and data analysis. At the knowledge level, frontier AI systems have excelled on real-world software engineering bug fixes and on graduate-level math and science questions that are difficult even for highly trained humans (Jimenez et al., 2024; Rein et al., 2024; Glazer et al., 2024; Phan et al., 2025). More importantly for scientific workflows, agentic systems promise a form of "end-to-end" execution: command-line coding agents such as Codex and Claude Code can read and modify codebases, install packages, run experiments, analyze outputs, create visualizations, and produce written summaries from high-level instructions (OpenAI, 2025; Anthropic, 2026; Merrill et al., 2026). As model-improvement techniques ranging from fine-tuning to self-evolving agents continue to develop, evaluation has become the next bottleneck for further progress.

Agent benchmarks for coding, data analysis, and science. Early code-generation benchmarks such as HumanEval and MBPP focus on synthesizing short programs that pass unit tests (Chen et al., 2021; Austin et al., 2021), while repository-level benchmarks such as SWE-bench evaluate agents on real GitHub issues with regression-test checking (Jimenez et al., 2024). These settings have been central to measuring progress because they admit relatively clear automated verification. More recent benchmarks extend evaluation to data analysis, machine-learning (ML) engineering, and scientific workflows: ScienceAgentBench constructs data-driven discovery tasks from peer-reviewed publications (Chen et al., 2025b); DSBench, BLADE, MLAgentBench, MLE-bench, AutoKaggle, and AutoML-Agent target realistic data-science, ML experimentation, competition, or AutoML pipelines (Jing et al., 2025; Gu et al., 2024; Huang et al., 2024; Chan et al., 2025; Li et al., 2024; Trirat et al., 2025); and DiscoveryBench studies whether agents can recover or verify scientific relationships from data (Majumder et al., 2025). Real scientific research, however, is more iterative and open-ended than these settings allow: task boundaries are less fixed, success is multi-dimensional, and ground truth can be expensive or ambiguous. Recent efforts have therefore leaned on expert curation, hierarchical rubrics, and bespoke challenge environments (Starace et al., 2025; Chen et al., 2025a; Wijk et al., 2025), with curators often prioritizing discrimination and difficulty so that future systems can hill-climb on older ones. We argue this "hill-climbing-first" framing should be complemented by a *user-centric workflow* in which scientists inspect what an

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

agent tried, decide whether the workflow was biologically sensible, and revise instructions accordingly.

Papers as executable evaluation artifacts. Closest to our approach are efforts that treat scientific papers as structured artifacts for agentic systems. PaperBench evaluates whether agents can replicate ICML 2024 ML papers by decomposing each replication into many author-informed, gradable subtasks (Starace et al., 2025). Paper2Agent takes a complementary direction, converting papers and their codebases into interactive agents that expose paper-specific methods as callable tools (Miao et al., 2025). Here we use benchmark papers more modestly: as sources of executable, hidden-scored tasks whose traces can be inspected by domain experts. The recipe applies broadly to benchmark papers whose input–output structure and evaluation criteria are clearly described, using papers not as deployable assistants or full-paper replication targets, but as blueprints for hidden, executable, domain-specific evaluations. We illustrate the idea using a recent benchmark of spatial transcriptomics alignment methods (Yan et al., 2026), compiling a subset of the original tasks spanning diverse tissues, platforms, and evaluation metrics. Focusing on one task family lets us examine failure modes that simpler multiple-choice benchmarks rarely expose, while leveraging the author team's background in computational genomics to evaluate method soundness. *We do not claim that spatial alignment exhausts scientific agent evaluation; rather, we view it as a concrete case study*, with replication in other domains needed to assess generalizability.

Benchmark validity, traces, and scientific shortcuts. Agentic benchmarks introduce validity challenges that are less pronounced in static question-answering settings. For instance, the Agentic Benchmark Checklist (Zhu et al., 2025) distinguishes *task validity*—whether success requires the intended capability being measured—from *outcome validity*, which concerns whether the scoring procedure correctly separates successful outputs from failures. In scientific-agent settings, preserving *task validity* requires guarding against leakage in the input data, overfitting to validation logic, or simple heuristics that learn correlational rather than scientific targets, such as approximating biological variation using technical variation from experimental batches. Drawing on existing best practices (Balloccu et al., 2024; MacDiarmid et al., 2025; METR, 2025), we use a scorer hidden from the agent workspace and combine endpoint metrics with trace inspection and leakage audits. This allows us to assess not only whether an alignment output is accurate, but also how the agent arrived at its final solution.

Spatial Alignment. Spatial Alignment (SA) aims to reconstruct the three-dimensional molecular architecture of tissues from two-dimensional slices, such as spatial transcriptomics measurements, using computational techniques. This task is central to spatial biology because integrating multiple tissue slices can reveal 3D tissue organization and molecular interactions that are not directly accessible from individual 2D measurements. Formally, an SA task takes two or more slices with observed spatial coordinates and gene-expression profiles and asks for a transformation or correspondence map between them. This map should align spatially and molecularly similar regions while accounting for noise, tissue deformation, and biological variation across slices.

Recently, Yan et al. (2026) evaluated a diverse suite of leading SA methods on 295 alignment tasks spanning multiple datasets and technologies. Their benchmark assessed method accuracy, efficiency, usability, and robustness, as well as the downstream impact of alignment quality. We use this benchmark as the input and human-expert performance ceiling, since in practice we would rarely be able to exhaustively search over all popular and leading packages, methods, and parameter settings.

Contributions. Using spatial alignment as a case study, we (i) show that frontier command-line coding agents can be evaluated on executable tasks derived from a well-designed computational biology benchmark; (ii) find that current coding agents often prefer simple coordinate- or geometry-based heuristics when solving alignment tasks; and (iii) show that explicit prompting and access to specialized packages *do not automatically improve performance*. Even with a virtual coding environment and packages installed, agent performance can degrade because of failed calls, brittle package integration, and internal proxy metrics only weakly aligned with the gold-standard metric.

Given these findings, we advocate that intermediate artifacts—agent traces and logs—should be shared as diagnostic signals for understanding where scientific agents succeed (Kirgis et al., 2026), where they switch to (sometimes surprisingly performant) heuristics, and how future user-in-the-loop scientific benchmarks should be designed.

2. Methods

We convert paper-derived spatial-alignment benchmarks into executable hidden-evaluation tasks for coding agents. The central object is a fixed task contract: the agent receives public inputs and an output schema, while labels, held-out expression targets, gold coordinates, metric definitions, and scorer metadata remain hidden. We then vary only the support available to the agent and evaluate all submissions with the same hidden scorer. This design separates three questions: which tasks can be reconstructed reproducibly from the paper, whether additional paper-derived support improves agent performance, and which failures arise from

scientific alignment rather than operational execution.

Terminology and abbreviations. SABench denotes the source spatial-alignment benchmark (Yan et al., 2026); the three execution conditions are Basic, Package-aware (PA), and Full + prior (FPP). We use large language model (LLM), command-line interface (CLI), application programming interface (API), mean absolute error (MAE), Pearson correlation coefficient (PCC), structural similarity index (SSIM), mutual information (MI), intersection over union (IoU), principal component analysis (PCA), and iterative closest point (ICP). Dataset-family labels follow the source benchmark: dorsolateral prefrontal cortex (DLPFC), mouse hypothalamic preoptic region (MHPR), primary motor cortex (MOp), whole mouse brain (WMB), adult mouse brain coronal (AMBC), squamous cell carcinoma (SCC), and breast cancer (BCA). Technology and file abbreviations use standard meanings: multiplexed error-robust fluorescence in situ hybridization (MERFISH), Common Coordinate Framework (CCF), comma-separated values (CSV), nearest neighbor (NN), and next-generation sequencing (NGS); Open-ST is kept as the platform name used by the source benchmark.

2.1. Benchmark construction

Benchmark construction defines the fixed task contract. We compiled 40 tasks from the 295 task definitions in (Yan et al., 2026). This 40-task set is not a random subset: the original benchmark publication provides links to the source datasets, but some task definitions could not be reconstructed reproducibly because intermediate consolidation files or derived assets were unavailable. We therefore report the common subset of 40 tasks that we could reconstruct with high confidence (Table 1).

For each task, we separate assets into an agent-facing workspace and scorer-only files. The agent-facing workspace contains the task prompt, processed expression matrices, observation identifiers, observed coordinates, and the required output schema. Scorer-only assets include labels, held-out marker genes, evaluation splits, 3D or atlas coordinates, paper-result metadata, and metric-profile information. Final scoring is invoked only by the harness *after* the agent exits.

2.2. Agent execution

We fixed the task contracts and hidden scorers and evaluated agents under three execution conditions that differ in prompt content, workspace setup, package access, and method context. In particular, we were interested in whether additional nudges to use benchmark-derived insights and packages would encourage the use of specific alignment packages and improve end-to-end agent performance.

Following recent agent benchmarks for computational tasks (Nair et al., 2026), all reported runs use Codex CLI with model `gpt-5.3-codex` and reasoning effort set to `xhigh`. The runner uses the Codex approval-bypass mode to avoid repeated human check-ins during autonomous task execution; the exact flag is documented in the prompt-template appendix.

For each dataset and execution condition, we fix the data splits and scorer settings and run three independent replicates. This allows us to characterize variability in agent performance due to LLM stochasticity. Replicate runs use independent workspaces. After the agent completes, a separate evaluation harness runs the leakage audit and passes the submitted files to the hidden scorer.

2.3. Agent evaluation

After an agent exits, the hidden scorer evaluates only the submitted coordinate tables. The scorer first checks the task contract: each required output file must be present, have the expected schema, preserve the required observation identifiers, and contain finite coordinates. Missing files, schema errors, row-identity mismatches, and non-finite coordinates are treated as failed submissions and assigned a score of zero in end-to-end analyses.

Valid submissions are scored with the metric profile fixed for that task. We follow the same broad evaluation logic as SABench (Yan et al., 2026): tasks without direct gold coordinates are evaluated using expression agreement and label or landmark consistency between aligned slices, whereas tasks with hidden gold coordinates also report coordinate error. Our primary endpoint for cross-task agent comparison is a higher-is-better composite over the metrics available for that task. MAE is reported separately because it is lower-is-better and is not on the same scale as the similarity metrics. The metric families include expression agreement, label or landmark alignment, gold-coordinate recovery, and diagnostic geometry; details are summarized in Table 5.

For each valid run, the scorer evaluates all applicable higher-is-better metrics in the task profile. Metrics computed on consecutive slice pairs are first averaged across pairs. The run-level composite score is then the unweighted average of the available higher-is-better metrics, $s = |\mathcal{H}|^{-1} \sum_{j \in \mathcal{H}} \bar{m}_j$, where $\mathcal{H}$ is the set of higher-is-better metrics available for the task, excluding MAE, and $\bar{m}_j$ is metric j after any averaging across slice pairs. This composite is computed separately for each task, execution condition, and replicate run. We note that the score s is used as a high-level aggregate metric for comparing agent runs across conditions and tasks. We still report individual metrics, which are easier to interpret, e.g., MAE is reported whenever the gold-standard coordinates are available.

Finally, beyond scorer outputs, we analyze run traces and

Table 1. Summary of the 40 tasks used for agent execution and evaluation.

Tier	Count	Task families / targets	Role in the benchmark
T1 exact paper table	2	DLPFC and MHPR	Direct comparison with paper-reported method tables.
T2 paper-adjacent metric	5	DLPFC robustness, Flysta3D, Open-ST	Same-family or direct MAE comparisons with task-construction caveats.
T3 local hidden gold	22	Breast cancer, mouse brain, MOp, WMB, Open-ST variants	Main hidden-evaluation tasks when no exact paper table is available.
T4 held-out expression	10	Breast cancer, AMBC/cross-platform, SCC/MHPR, mouse brain variants	Generalization tests using withheld expression targets.
T5 diagnostic/local	1	Breast-cancer resolution diagnostic	Stress-test and diagnostic task.

Table 2. Agent execution conditions.

Condition	What is provided to the agent
Basic	Task prompt only; no paper-method context and no prebootstrapped method environments.
Package-aware	Paper-method names and package suggestions in the prompt; the agent installs and probes packages itself, with no prebootstrapped method environments.
Full + prior	Prebootstrapped method-package environments plus SABench-derived high-level method priors by task family. These priors summarize which paper methods were favored for related task families, but they do not expose hidden labels, held-out expression targets, or scorer outputs.

generated code to characterize agent behavior with LLM assistance. Specifically, we use an independent LLM call to extract failure stages, error messages, package-use signals, and method-use signals from observable evidence.

3. Findings

Throughout this section, we abbreviate the two execution conditions in Table 2 as PA (Package-aware) and FPP (Full + prior).

3.1. More prompt context increased package use, but not aggregate accuracy

We first checked whether the prompt and workspace changes actually changed agent behavior. They did. As intended, giving agents more benchmark-specific context made them more likely to use spatial-alignment packages in their final solutions. FPP, which includes specific package hints as well as prebuilt virtual environments with installed packages, produced the strongest observable package-use signals: PASTE2 in 72% of runs, PASTE in 68%, Spateo in 54%, and SPACEL in 43% (Figure 1). By comparison, PA showed a narrower package footprint, with roughly 48% PASTE use, less than 5% STAligner use, and little meaningful use of other packages. Basic showed almost no package use, except for roughly 20% PASTE calls, likely because PASTE had a simple API and appeared early in the prompt.

The more surprising result is that this increase in package use did not translate into uniformly better scores. In Figure 1, each point represents one task's mean composite score across three replicate agent runs. The condition with the strongest package-use signal, FPP, had the lowest average score. On the 40-task common subset, Basic achieved the highest mean composite score (0.428, 95% bootstrap CI [0.377, 0.486]), followed closely by PA (0.421 [0.370, 0.478]). FPP was lower (0.361 [0.314, 0.416]). Paired task-level comparisons tell the same story: FPP underperformed Basic by -0.067 (95% paired-bootstrap CI $[-0.113, -0.028]$, Wilcoxon signed-rank $p = 0.0021$) and PA by -0.059 ($[-0.105, -0.017]$, $p = 0.021$), while PA and Basic were not significantly different (-0.008 $[-0.043, +0.029]$, $p = 0.48$).

The same pattern is visible across tasks. Figure 2 draws each task as a trajectory through Identity, Basic, PA, and FPP. We stratify tasks by the Identity baseline, where Identity means submitting the input coordinates unchanged. Easy, Medium, and Hard tasks are defined by tertiles of the Identity composite score. Although individual tasks vary substantially, the cohort mean declines from Basic to FPP. This drop is especially visible on Easy tasks, where some FPP solutions are not meaningfully better than a no-op submission.

3.2. Metric-specific results show the same pattern

One possible concern is that the composite score could be driven by only a few metrics. We therefore also examined the individual metric families. The full per-metric breakdown is reported in the appendix. Across the metrics in the profile, FPP is lower on average than Basic on every individual metric. The largest negative differences appear for gene cosine similarity, matching accuracy, and region-grid IoU. This suggests that the aggregate trend is not an artifact of one unusually noisy or unbounded metric.

3.3. Why package use and metric outcomes diverged

To understand why higher package use did not reliably improve scores, we inspected the observable search process of the coding agents, especially under PA and FPP. The logs

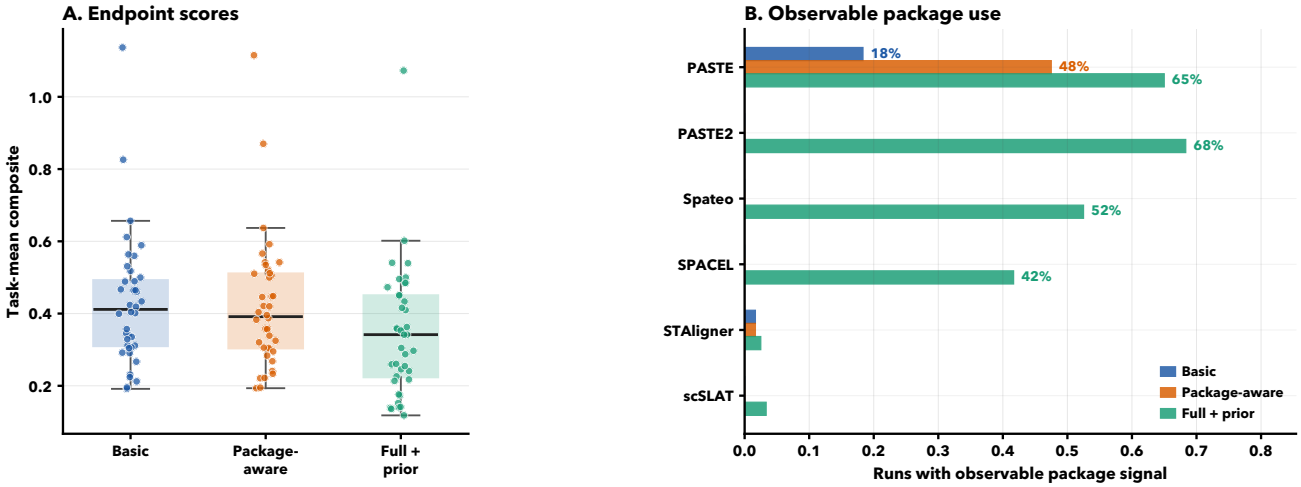

Figure 1. Endpoint and behavior summaries for the 40-task common subset. Left: per-task mean composite scores. Right: observable package-use signals. Full + prior produced the strongest package-use signal, but this behavioral shift did not translate into higher benchmark scores.

show that additional context changed the search process, but not always in ways that matched the task.

Overall, FPP had the strongest package-oriented behavior: 93% of runs probed packages, 75% had a successful method-environment command, and about 80% of scored runs submitted package-backed final solvers. Basic agents more often used inspect-then-custom-solution or package-attempt-to-custom-fallback strategies. On many tasks, these lighter strategies were competitive with, and sometimes better than, package calls run with weakly chosen parameters.

The Identity baseline makes one failure mode especially visible. Identity submits the public input coordinates unchanged; therefore, a run that scores below Identity has actively degraded an input that was already partly aligned. On the 39 scored tasks, Basic falls below Identity on 2 tasks, PA on 2 tasks, and FPP on 6 tasks. The pattern is sharper on the 13 Easy-tier tasks, where Identity is already relatively strong: FPP falls below Identity on 5 tasks, compared with 2 for Basic. Figure 3 shows two representative cases, SA-004 and SA-014. In both, the input coordinates already give a useful alignment, while agent-selected transformations move the slices away from that reference.

For the subset of tasks with hidden gold coordinates, we also inspect MAE separately. MAE is available for five of the 40 tasks. On these tasks, the advantage of Basic is less clear: differences are often within run-to-run noise, and the lowest mean MAE for a task always comes from either PA or FPP, with FPP best on three of the five tasks. This suggests that results based on gene- and neighborhood-based metrics should be interpreted with some caution. When hidden gold coordinates are available, MAE is a more direct coordinate recovery target, although it is task-scale dependent and not directly comparable across all task families. Figure 4 shows representative hidden-coordinate tasks with the scorer-only gold frame included as a visual reference.

We next asked whether agent outputs exceeded simple coordinate-only probes. These probes are useful because many agent-written solvers converged to the same class of geometry operations when package context was absent. All three probes operate sequentially: the first slice is kept fixed, and each later slice is mapped to the previously aligned slice using only observed coordinates, with no gene expression, labels, hidden gold, or package-specific method code. The *center/scale* probe translates the moving slice to the reference centroid and rescales it so that the median distance from the centroid matches the reference. The *PCA-similarity* probe also aligns the principal coordinate axes, evaluates the four possible axis sign flips, and keeps the candidate with the largest bounding-box overlap. The *trimmed ICP* probe starts from the PCA-similarity and center/scale candidates, iteratively matches nearest-neighbor coordinates, fits a two-dimensional similarity transform to the closest 85% of matched pairs, and keeps the refinement with the smallest nearest-neighbor distance.

These probes are not intended as strong scientific methods; they ask whether an agent is doing more than simple geometry. On the 38 tasks where they are applicable, center/scale averages a composite score of 0.311, PCA-similarity averages 0.366, and trimmed ICP averages 0.414. On the same tasks, Basic averages 0.423, PA averages 0.415, and FPP averages 0.357. Thus the strongest coordinate-only probe nearly matches PA and exceeds FPP. Package-heavy strategies can outperform these probes on individual tasks, but in aggregate the variance introduced by package APIs, parameter choices, and recovery loops lowers their expected performance.

Finally, we checked whether the benchmark was dominated by a single package family, which would suggest that a more

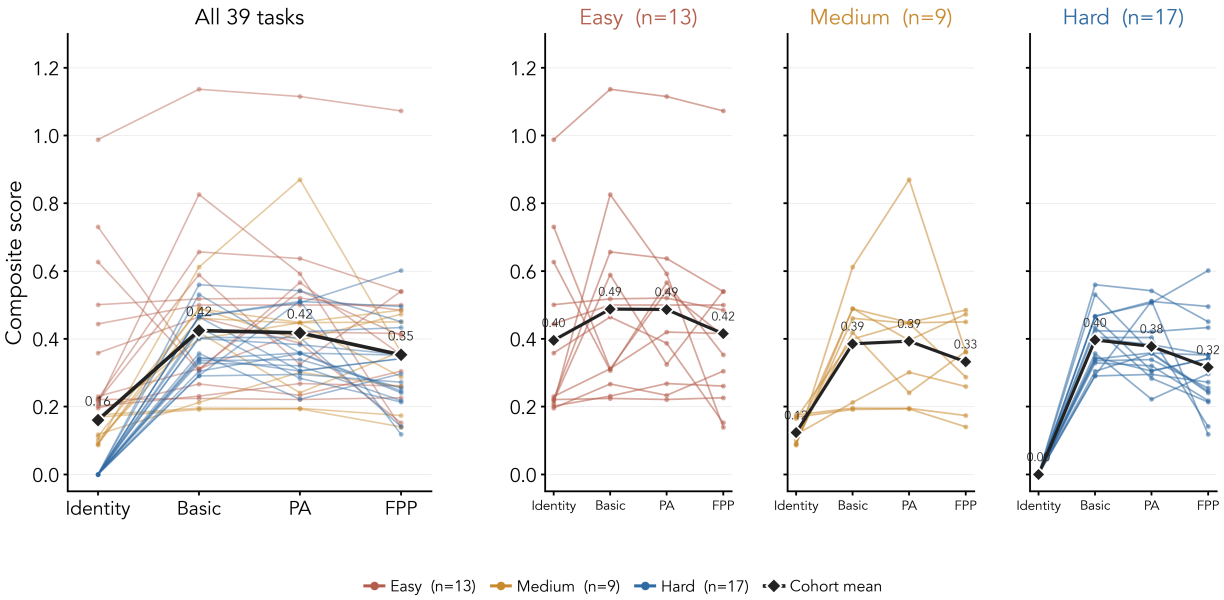

Figure 2. Per-task trajectories on the composite score. Each thin line is one task colored by Identity tier; black diamonds denote cohort means. Left: all 39 scored tasks. Right: the same trajectories faceted by tier. Tertile cutoffs are Hard at Identity ≤ 0.00 ($n = 17$), Medium at $0 <$ Identity ≤ 0.18 ($n = 9$), and Easy at Identity > 0.18 ($n = 13$). Axis labels use PA for Package-aware and FPP for Full + prior. In every tier, Full + prior ends below Basic.

focused package prompt might solve the problem. We found no such universal winner. Rigid or affine custom geometry performs best on some serial-alignment families; Spateo and GPSA are stronger on SCC; PASTE helps in some BCA and AMBC settings; and on DLPFC and MHPR, package and rigid means often cluster closely. The benchmark therefore rewards local method selection, not simply increased package availability (Table 8).

3.4. Concrete examples and trace inspection

We visualize two paired examples to separate useful package context from harmful package pressure. Here FPP denotes Full + prior, and the task labels refer to the compiled spatial-alignment task IDs. The largest FPP win over Basic is SA-003, a DLPFC pair in which one slice's coordinates were coarsely scrambled during task construction. Basic agents produced inconsistent custom geometry, whereas FPP agents converged on Spateo and PASTE2 alignments. This is the setting where benchmark-derived method context helps: the task calls for the kind of nontrivial transformation that spatial-alignment packages are designed to solve.

The largest FPP loss is SA-006, a clean serial alignment of two mouse-brain sections. Basic agents solved the task with short rigid or affine custom solvers. FPP agents instead followed package-backed workflows and submitted alignments that were close in broad structure but misoriented enough to reduce gene-grid and landmark scores. Here the additional method context pulled the agent away from a simple sufficient solution. The paired visualization for these two cases is included in Figure 7.

Trace inspection shows how these outcomes arise. We analyze only observable run artifacts: prompts, status messages, shell commands, command outputs, generated code, scorer logs, submitted files, and workspace snapshots. We do not inspect hidden model reasoning. These traces show that FPP often changed the search process in the intended direction, but also increased long package loops, environment failures, and prior-driven submissions.

The trace vignettes in Table 3 make these process differences more concrete. They show whether the agent wrote a valid submission before exploring, recognized a package stall, patched data-contract issues, switched to a lighter fallback, or lost credit after an operational failure. These are not reasoning transcripts; they are summaries of externally visible run evidence.

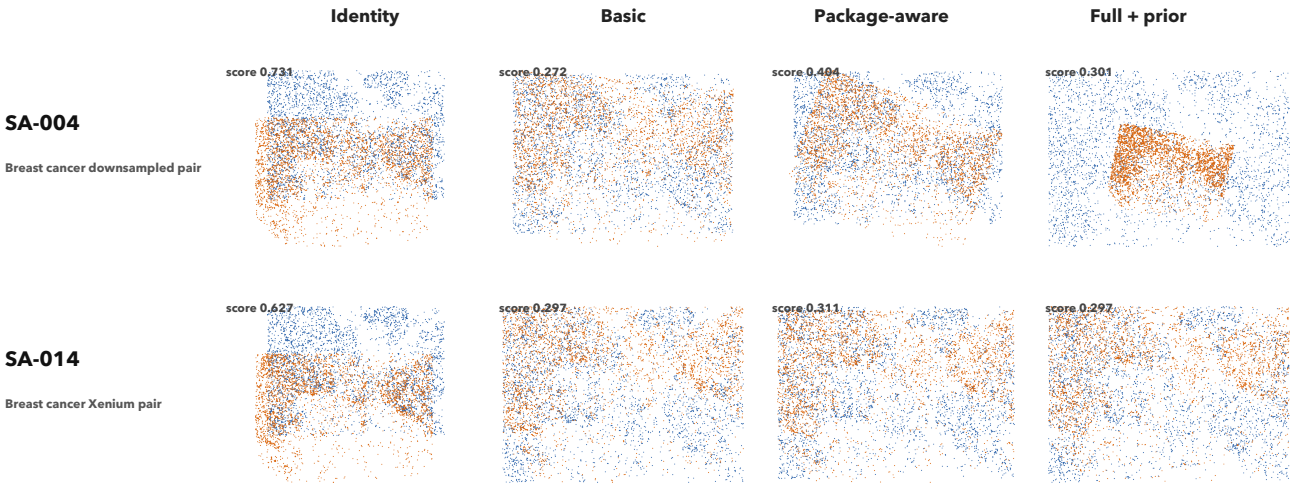

Figure 3. Examples where transforming an already useful input degrades performance. Columns show the Identity reference and representative valid submissions from Basic, Package-aware, and Full + prior. Identity denotes the public input coordinates submitted without transformation; it is a diagnostic reference, not a coordinate-level ground truth. For SA-004 and SA-014, the hidden scorer uses held-out expression and region/landmark agreement rather than a known target coordinate map.

Table 3. Concrete observable trace vignettes. Short excerpts summarize visible run artifacts; exact longer excerpts are reported in Section E.

Trace family	Observable excerpt	Process signal
Baseline insurance	SA-001 Basic: expression-guided transform unstable; one trial hit a runtime failure.	**Valid baseline first**; risky refinement did not break the submission.
Slow package fallback	SA-001 Basic: two long-running `paste` jobs were terminated; validated CSVs remained unchanged.	**Package attempt stalled**; fallback preserved the run.
Geometry + expression check	SA-003 Basic: the same transform was best by geometry and expression consistency.	**Geometry was checked against expression** before submission.
Large multi-slice execution	SA-020 Basic: all 33 CSVs were written after serial alignment.	**Complete multi-slice output** was validated after a custom solve.
Package-backed selection	SA-008 Full + prior: `spateo` beat a custom rigid fallback by input-only diagnostics.	**Package result was selected by visible diagnostics**.
Package error recovery	SA-006 Full + prior: unequal gene sets triggered an intersect/reorder patch.	**Package data-contract issues were patched**, but the rerun remained slow.
Provider disconnect after work	SA-105 Full + prior: `PASTE2` was still running; a completed `spateo` submission existed.	**Endpoint failure hid completed work**; provider failure zeroed the run.

Finally, we audited deterministic trace labels with an LLM-assisted review using only observable evidence. The review inputs included scores, Identity baselines, submitted approach family, package-environment successes and failures, command excerpts, code excerpts, and report files. We use only strict consensus labels as process diagnostics, not as ground truth. Table 4 summarizes the two-judge audit over the Full + prior traces. The judges did not see hidden gold labels, private reasoning text, or scorer internals; they saw only observable run artifacts and a fixed label rubric.

The audit supports the same qualitative picture as the endpoint results: FPP often produces package-backed submissions that beat Identity, but its failures are mechanistically heterogeneous. Among strict-consensus FPP traces, 64 of 120 runs were package-backed submissions that beat Identity. The remaining failures include 17 package-backed submissions no better than Identity, 11 operational failures, 12 probe-and-bail or self-evaluation fallbacks, and 2 runs with no package attempt. This audit is not used as a scoring signal, but it helps separate failure modes that look similar under the endpoint metric alone.

Table 4. LLM judge audit over 120 Full + prior traces. Counts use only observable artifacts and strict two-judge consensus.

Audit quantity	Count	Interpretation
Strict consensus labels	106/120	Used for subtype summaries.
Explicit judge disagreements	10/120	Routed to review.
Missing Identity reference	2/120	Package-vs.-Identity unavailable.
Missing second-judge label	2/120	Routed to review.
Package-backed beats Identity	64/120	Package use clears no-op.
Package-backed no better than Identity	17/120	Over-engineering case.
Infrastructure/scorer failure	11/120	End-to-end zero rows.
Package probing followed by baseline	12/120	Probe-and-bail or self-eval fallback.
No package attempt	2/120	Pure baseline behavior.

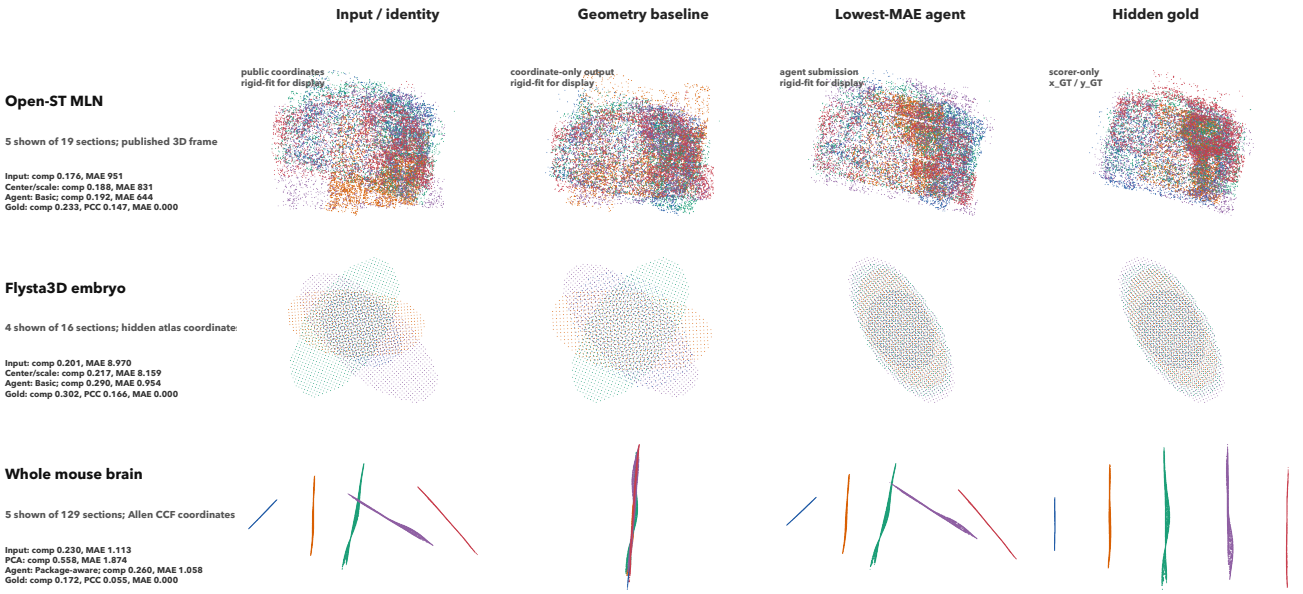

Figure 4. Examples with hidden coordinate gold. Rows show three tasks with scorer-only 3D or atlas-coordinate references; columns compare the public input, the best deterministic coordinate-only baseline, the valid agent submission with the lowest MAE, and the hidden gold frame. Submitted coordinates are rigid-fitted to the hidden frame for display only. The gold column was never visible to the agent; its MAE is zero by construction, while its displayed composite can differ because the composite excludes MAE and averages higher-is-better expression or label metrics.

4. Discussion and conclusion

Key findings. This case study sits between toy coding benchmarks and open-ended scientific discovery: the agent must choose methods, adapt package APIs, manage computational limits, and satisfy a hidden scientific scorer. The central finding is that more domain affordance did not reliably help. Full + prior increased package probing and package-backed submissions, yet produced the lowest aggregate composite score. Trace evidence explains why this negative result is informative rather than merely disappointing. Some tasks benefited from package context, especially when the input required a nontrivial spatial transformation. Others rewarded a no-op, a rigid transform, or a simple coordinate-only probe, and the package-rich regime often overrode those local signals. Endpoint scores identify which regime won; traces show whether the workflow would survive domain-expert inspection.

Limitations. The empirical claims are bounded to one benchmark family: SABench-style spatial transcriptomics alignment. The three regimes also bundle prompt content, package access, prebuilt environments, and method priors, so we do not isolate which affordance caused the degradation. The baseline probes are intentionally modest; label-aware Procrustes, expression-nearest-neighbor, and mutual-nearest-neighbor baselines remain useful additions. Paper comparisons are tiered because only two tasks have exact extracted SABench tables aligning closely with our local scorer. Finally, the study uses one hosted coding-agent configuration with three replicates per task, so cross-model and reasoning-budget sensitivity remain open.

Future work. A promising next step is to test whether instructing agents to assess task-local evidence before transforming already-useful coordinates improves results, moving toward more interactive and capable coding agents. More broadly, our trace-inspection approach should be replicated beyond spatial alignment, especially in domains where agents must construct proxy feedback under delayed or hidden evaluation rather than optimize against an immediately visible score.

Conclusion. Benchmark papers can be more than static comparisons among human-authored methods. When compiled into executable tasks with hidden evaluation, fixed output contracts, trace capture, and leakage controls, they become diagnostics for how AI agents actually conduct scientific work—and for what a scientist would need to inspect before trusting the result. Our spatial genomics case study shows both promise and friction: agents can use real packages and sometimes improve task-specific outcomes, but added scientific context can also create action bias, brittle package loops, and lower aggregate accuracy. The broader lesson is that scientific-agent scaffolds should help agents compare external priors with task-local evidence—and should make that evidence trail visible to the user.

References

Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2026. Accessed 2026-05-06.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Balloccu, S., Schmidtova, P., Lango, M., and Dušek, O. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, 2024.

Chan, J. S., Chowdhury, N., Jaffe, O., Aung, J., Sherburn, D., Mays, E., Starace, G., Liu, K., Maksin, L., Patwardhan, T., Madry, A., and Weng, L. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095.

Chen, H., Xiong, M., Lu, Y., Han, W., Deng, A., He, Y., Wu, J., Li, Y., Liu, Y., and Hooi, B. MLR-Bench: Evaluating AI agents on open-ended machine learning research. *arXiv preprint arXiv:2505.19955*, 2025a. URL https://arxiv.org/abs/2505.19955.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, Z., Chen, S., Ning, Y., Zhang, Q., Wang, B., Yu, B., Li, Y., Liao, Z., Wei, C., Lu, Z., et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025b. URL https://arxiv.org/abs/2410.05080.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., de Oliveira Santos, E., Järviniemi, O., Barnett, M., Sandler, R., Vrzala, M., Sevilla, J., Ren, Q., Pratt, E., Levine, L., Barkley, G., Stewart, N., Grechuk, B., Grechuk, T., Enugandla, S. V., and Wildon, M. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024. URL https://arxiv.org/abs/2411.04872.

Gu, K., Shang, R., Jiang, R., Kuang, K., Lin, R.-J., Lyu, D., Mao, Y., Pan, Y., Wu, T., Yu, J., et al. BLADE: Benchmarking language model agents for data-driven science. *arXiv preprint arXiv:2408.09667*, 2024. URL https://arxiv.org/abs/2408.09667.

Huang, Q., Vora, J., Liang, P., and Leskovec, J. MLAgentBench: Evaluating language agents on machine learning experimentation. In *International Conference on Machine Learning*, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Jing, L., Huang, Z., Wang, X., Yao, W., Yu, W., Ma, K., Zhang, H., Du, X., and Yu, D. DSBench: How far are data science agents from becoming data science experts? In *International Conference on Learning Representations*, 2025.

Kirgis, P., Kapoor, S., Rabanser, S., Nadgir, N., Ududec, C., Dubois, M., Allaire, J., Stosz, C., Hobbhahn, M., Steinhardt, J., and Narayanan, A. Log analysis is necessary for credible evaluation of ai agents, 2026. URL https://arxiv.org/abs/2605.08545.

Li, Z., Zang, Q., Ma, D., Guo, J., Zheng, T., Liu, M., Niu, X., Wang, Y., Yang, J., Liu, J., et al. AutoKaggle: A multi-agent framework for autonomous data science competitions. *arXiv preprint arXiv:2410.20424*, 2024. URL https://arxiv.org/abs/2410.20424.

MacDiarmid, M., Wright, B., Uesato, J., Benton, J., Kutasov, J., Price, S., Bouscal, N., Bowman, S., Bricken, T., Cloud, A., Denison, C., Gasteiger, J., Greenblatt, R., Leike, J., Lindsey, J., Mikulik, V., Perez, E., Rodrigues, A., Thomas, D., Webson, A., Ziegler, D., and Hubinger, E. Natural emergent misalignment from reward hacking in production RL. *arXiv preprint arXiv:2511.18397*, 2025.

Majumder, B. P., Surana, H., Agarwal, D., Bhavana Dalvi Mishra, Meena, A., Prakhar, A., Vora, T., Khot, T., Sabharwal, A., and Clark, P. DiscoveryBench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vyflgpwfJW.

Merrill, M. A., Shaw, A. G., Carlini, N., Li, B., Raj, H., Bercovich, I., Shi, L., Shin, J. Y., Walshe, T., Buchanan, E. K., et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.11868.

METR. Recent frontier models are reward hacking. https://metr.org/blog/2025-06-05-recent-reward-hacking/, 2025. Published June 5, 2025.

Miao, J., Davis, J. R., Zhang, Y., Pritchard, J. K., and Zou, J. Paper2Agent: Reimagining research papers as interactive and reliable AI agents. *arXiv preprint arXiv:2509.06917*, 2025. URL https://arxiv.org/abs/2509.06917.

Nair, S., Gunsalus, L., Orcutt-Jahns, B., Rossen, J., Lal, A., De Donno, C., Çelik, M. H., Fletez-Brant, K., Xie, X., Bravo, H. C., and Eraslan, G. Agentic systems are adept at solving well-scoped, verifiable problems in computational biology. *bioRxiv*, 2026. doi: 10.64898/2026.04.06.716850. URL https://www.biorxiv.org/content/early/2026/04/09/2026.04.06.716850.

OpenAI. Introducing codex. https://openai.com/index/introducing-codex/, 2025. Accessed 2026-05-06.

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025. URL https://arxiv.org/abs/2501.14249.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2311.12022.

Starace, G., Jaffe, O., Sherburn, D., Aung, J., Chan, J. S., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., Heidecke, J., Glaese, A., and Patwardhan, T. PaperBench: Evaluating AI's ability to replicate AI research. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2504.01848.

Trirat, P., Jeong, W., and Hwang, S. J. AutoML-Agent: A multi-agent LLM framework for full-pipeline AutoML. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60099–60146. PMLR, 2025. URL https://proceedings.mlr.press/v267/trirat25a.html.

Wijk, H., Lin, T., Becker, J., Jawhar, S., Parikh, N., Broadley, T., Chan, L., Chen, M., Clymer, J. M., Dhyani, J., Ericheva, E., Garcia, K., Goodrich, B., Jurkovic, N., Kinniment, M., Lajko, A., Nix, S., Sato, L. J. K., Saunders, W., Taran, M., West, B., and Barnes, E. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Yan, Y., Gu, T., Sun, C., Zhang, Y., Cui, Y., Lin, S., Zou, Q., Du, Y., Han, C., Kang, K., Li, S., Zhao, Y., Lin, Z., Yuan, Z., and Qian, B.-Z. Benchmarking alignment methods for spatial transcriptomics data. *Nature Computational Science*, 2026. doi: 10.1038/s43588-026-00977-z. URL https://doi.org/10.1038/s43588-026-00977-z.

Zhu, Y., Jin, T., Pruksachatkun, Y., Zhang, A., Liu, S., Cui, S., Kapoor, S., Longpre, S., Meng, K., Weiss, R., et al. Establishing best practices for building rigorous agentic benchmarks. *arXiv preprint arXiv:2507.02825*, 2025. URL https://arxiv.org/abs/2507.02825.

A. Metric families

Table 5. Metric families used by the hidden scorer. The main endpoint is the higher-is-better composite; MAE is reported separately for tasks with hidden gold coordinates. The full task-by-task metric-profile mapping is included in the released task manifests.

Metric family	Scoring role
Expression agreement	Compares marker-gene expression between matched grid cells in adjacent aligned slices; examples include gene PCC, gene cosine, gene SSIM, and gene mutual information. Higher values indicate stronger local expression agreement.
Label / landmark alignment	Compares hidden region labels after alignment using nearest-neighbor matching and region-level geometric overlap; examples include matching accuracy and region-overlap ratio. Higher values indicate better regional correspondence.
Gold-coordinate recovery	Computes MAE between submitted coordinates and held-out gold coordinates after rigid alignment of the submitted frame to the gold frame. Lower values indicate more accurate coordinate recovery.
Diagnostic geometry	Records local sanity checks, such as bounding-box overlap, used for diagnostic tasks and failure interpretation.

B. Task inventory

Table 6 lists all 40 tasks in the common subset with dataset family, scoring profile, slice count, and paper-comparability tier (T1 = exact paper metric table; T2–T3 = same family with caveats; T4 = held-out expression gold only; T5 = local diagnostic).

Table 6. Per-task inventory for the 40-task common subset.

#	Task ID	Dataset family	Metric profile	Slices	Tier
1	`sa-001-dlpfc-serial-align`	DLPFC	Gene+ label	4	T1
2	`sa-002-dlpfc-robust-perturb`	DLPFC	Gene+ label	2	T3
3	`sa-003-dlpfc-coarse-shuffled-pair`	DLPFC	Gene+ label	2	T2
4	`sa-004-bca-downsampled-pair-align`	Breast cancer	Gene	2	T4
5	`sa-005-bca-resolution-recovery`	Breast cancer	Res.	0	T5
6	`sa-006-mbsa-serial-align`	Mouse brain	Gene	2	T4
7	`sa-007-mbsp-serial-align`	Mouse brain	Gene	2	T4
8	`sa-008-ambc-xenium-multisection-serial-align`	AMBC/cross-platform	Gene	3	T4
9	`sa-009-mhpr-merfish-serial-align`	SCC/MHPR	Gene+ label	5	T1
10	`sa-010-ambc-visium-xenium-cross-platform`	AMBC/cross-platform	Gene	2	T4
11	`sa-011-scc-p2-serial-align`	SCC/MHPR	Gene	3	T4
12	`sa-012-scc-p5-serial-align`	SCC/MHPR	Gene	3	T4
13	`sa-013-ambc-visium-xenium5k-cross-platform`	AMBC/cross-platform	Gene	2	T4
14	`sa-014-bca-xenium-serial-align`	Breast cancer	Gene+ label	2	T3
15	`sa-015-ambc-visium-visiumhd-cross-resolution`	AMBC/cross-platform	Gene	2	T4
16	`sa-016-ambc-visium-starmapplus-cross-platform`	AMBC/cross-platform	Gene	2	T4
17	`sa-017-baristaseq-mpc-serial-align`	Mouse brain	Gene+ label	3	T3
18	`sa-018-starmap-mmpc-serial-align`	Mouse brain	Gene+ label	3	T3
19	`sa-019-flysta3d-e14-16-3d-serial-align`	Flysta3D	MAE	16	T2
20	`sa-020-mop-mouse1-atlas-serial-align`	MOp	Gene+ label	33	T3
21	`sa-021-mop-mouse2-atlas-serial-align`	MOp	Gene+ label	31	T3
22	`sa-022-openst-mln-serial-align`	Open-ST MLN	Gene+ label	19	T3
23	`sa-023-wmb-zhuang-abca1-129-section-3d-mae`	Breast cancer	MAE	129	T3
25	`sa-025-openst-mln-adjacent-02-03-3d-align`	Open-ST MLN	MAE	2	T2
26	`sa-026-openst-mln-early-window-02-06-3d-align`	Open-ST MLN	MAE	5	T2
27	`sa-027-openst-mln-full-19-slice-3d-align`	Open-ST MLN	MAE	19	T2
28	`sa-028-wmb-animal1-window5-serial-align`	WMB	Gene+ label	5	T3
29	`sa-029-wmb-animal1-window10-serial-align`	WMB	Gene+ label	10	T3
101	`sa-101-mop-mouse1sample1-serial-align`	MOp	Gene+ label	6	T3
102	`sa-102-mop-mouse1sample2-serial-align`	MOp	Gene+ label	5	T3
103	`sa-103-mop-mouse1sample3-serial-align`	MOp	Gene+ label	4	T3
104	`sa-104-mop-mouse1sample4-serial-align`	MOp	Gene+ label	6	T3
105	`sa-105-mop-mouse1sample5-serial-align`	MOp	Gene+ label	6	T3
106	`sa-106-mop-mouse1sample6-serial-align`	MOp	Gene+ label	6	T3
107	`sa-107-mop-mouse2sample1-serial-align`	MOp	Gene+ label	4	T3
108	`sa-108-mop-mouse2sample2-serial-align`	MOp	Gene+ label	6	T3
109	`sa-109-mop-mouse2sample3-serial-align`	MOp	Gene+ label	6	T3
110	`sa-110-mop-mouse2sample4-serial-align`	MOp	Gene+ label	5	T3
111	`sa-111-mop-mouse2sample5-serial-align`	MOp	Gene+ label	5	T3
112	`sa-112-mop-mouse2sample6-serial-align`	MOp	Gene+ label	5	T3

C. Supplementary endpoint results

Table 7. Per-metric breakdown across execution conditions. Means are over per-task replicate means on the largest subset where each metric is defined ($n = 39$ for gene metrics, $n = 25$ for landmark metrics). The Full + prior − Basic delta and paired Wilcoxon p are computed on the same per-task pairing as the headline comparison.

Metric (range, n)	Basic	PA	FPP	Δ	p
gene PCC ([-1,1], 39)	0.318	0.331	0.274	−0.044	0.22
gene cosine ([-1,1], 39)	0.711	0.691	0.609	−0.102	0.0002
gene SSIM ([-1,1], 39)	0.317	0.329	0.274	−0.044	0.23
gene MI ([0,∞), 39)	0.199	0.206	0.161	−0.039	0.10
matching acc ([0,1], 25)	0.521	0.473	0.430	−0.091	0.0018
region IoU ([0,1], 25)	0.548	0.498	0.429	−0.119	0.0003

Figure 5 shows the same result visually. The task-level trajectories differ from metric to metric, but the Full + prior endpoint remains below Basic across the panels. We therefore treat the composite as a compact index of the benchmark rather than as the sole source of evidence.

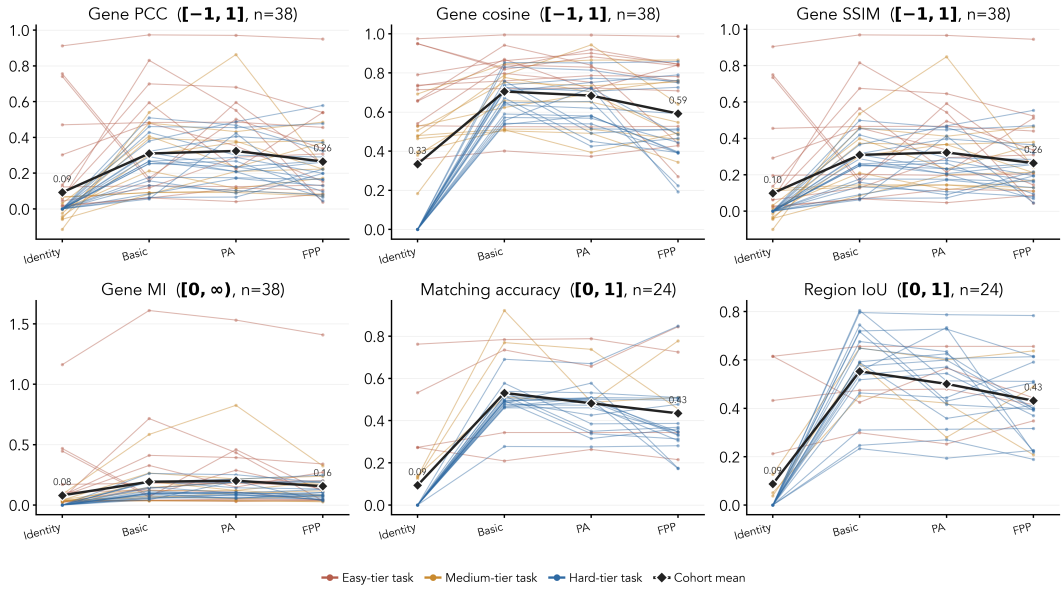

Figure 5. Per-task trajectories by individual metric. Identity tier is assigned once at the task level using composite-Identity tertiles and then carried into each per-metric panel. The Full + prior endpoint sits below Basic on every metric. Axis labels use PA for Package-aware and FPP for Full + prior.

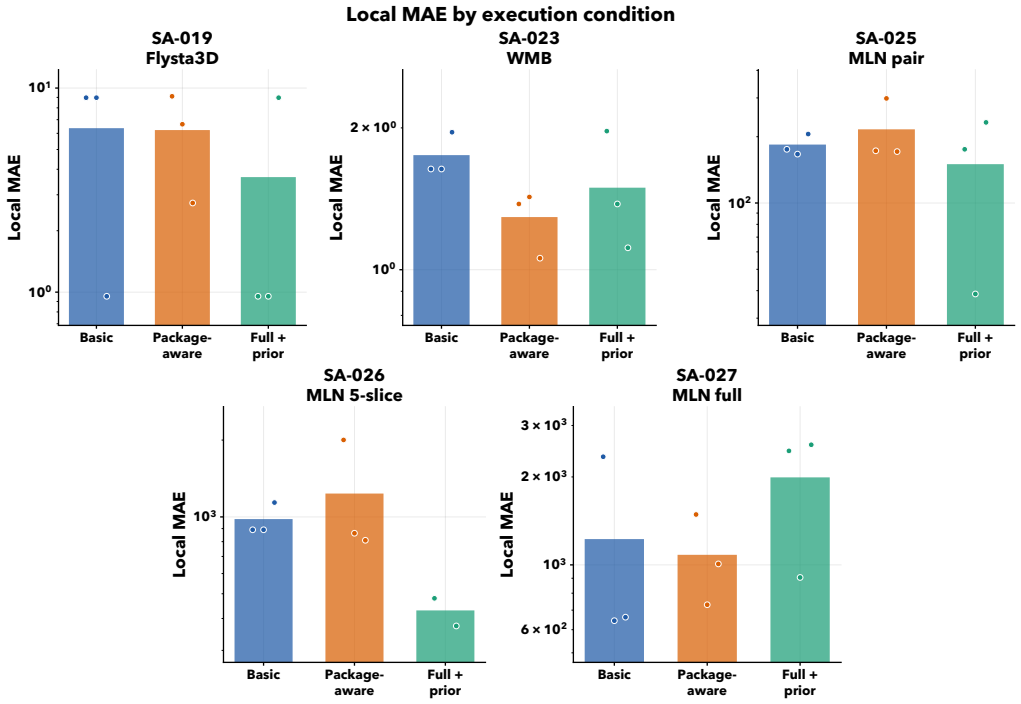

Figure 6. Local MAE comparison across execution conditions for tasks with hidden gold coordinates. Bars show condition means and points show individual runs. Lower is better. MAE is reported separately from the composite because it is lower-is-better and task-scale dependent.

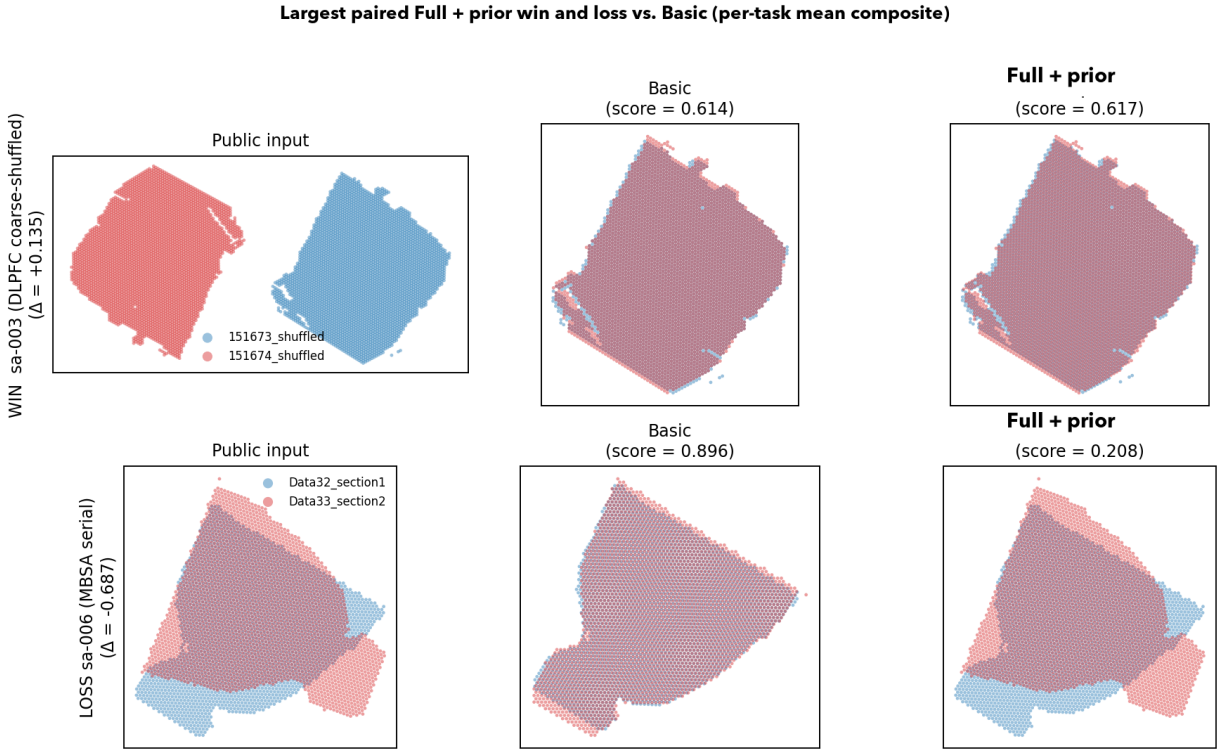

Figure 7. Largest paired Full + prior win and loss against Basic on the 40-task common subset. Each row is one task; columns show the public input coordinates and the highest-scoring Basic and Full + prior submissions. Top: SA-003 is a DLPFC coarse-shuffled pair where spatial-alignment packages help. Bottom: SA-006 is a clean serial-alignment task solved well by short custom geometry but degraded by package-driven alignment.

D. Paper-reference and diagnostic analyses

D.1. Additional diagnostics and paper-reference comparisons

The remaining analyses are useful for interpretation but more caveated than the main benchmark comparisons. We therefore treat them as supplementary diagnostics rather than as the primary result.

Paper comparisons require explicit comparability tiers. Only two tasks have exact extracted paper metric tables that align closely with the local task construction: DLPFC serial alignment and MHPR MERFISH serial alignment. Other source-benchmark results provide useful context but differ in metric definition, coordinate scale, or availability of raw per-method tables. Figure 8 overlays agent results with the published SABench method values for the exact direct-table cases.

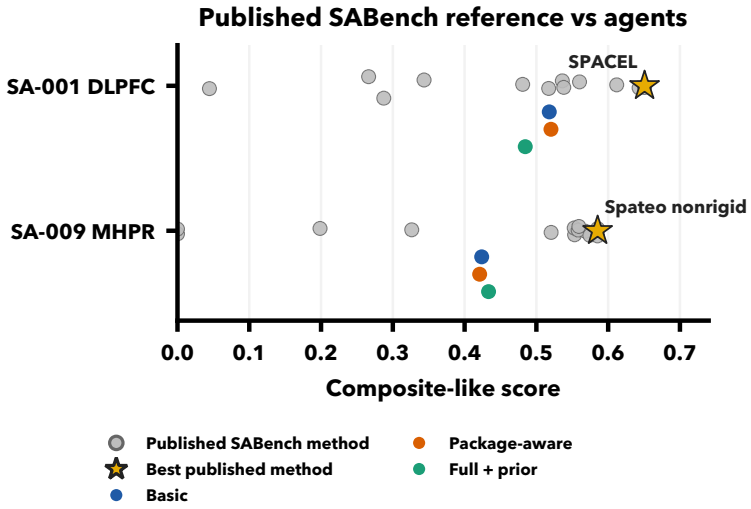

Figure 8. Paper-reported benchmark reference for the two exact direct-table cases. Gray points are published SABench method pipelines, gold stars are the best published method values, and colored points are agent execution-condition means.

Taken together, the supplementary analyses support the same interpretation as the main results. The issue is not simply that agents lack access to strong spatial-alignment methods. The harder problem is deciding when those methods are appropriate, when a simpler geometric solution is sufficient, and when the public input coordinates should be left nearly unchanged.

Table 8. Five representative tasks showing the spread of method-family outcomes. Cells are mean composite scores. Bold indicates the best method family in the row; italic indicates the Identity baseline when available; em-dash indicates no runs of that family for the task.

Task	Identity	rigid/affine	ICP	PASTE	PASTE2	Spateo	SPACEL	GPSA
`sa-006` (MB-serial)	*0.21*	**0.71**	—	—	0.21	—	—	—
`sa-014` (BCA Xenium serial)	*0.63*	—	—	0.63	—	0.31	—	—
`sa-011` (SCC)	*0.22*	0.70[†]	—	0.64	0.22	**0.70**	—	0.64
`sa-023` (WMB 3D MAE)	*0.23*	0.32	**0.65**	—	—	0.63	—	—
`sa-003` (DLPFC shuffled)	*0.00*	0.50	—	—	**0.61**	—	0.59	—

[†] `sa-011` has only $n = 1$ rigid/affine run; Spateo at $n = 2$ is the more reliable best. Across these examples, no single method family dominates.

Table 9. Additional diagnostics on the 40-task common subset. Values are computed after rerun de-duplication at the task level. Identity and coordinate-probe rows are behavioral diagnostics; for paired regime deltas, positive values favor the Full + prior regime.

Diagnostic	Value
Below Identity tasks	Basic 2/39; Package-aware 2/39; FPP 6/39
Easy-tier delta vs Identity (n=13)	Basic $+0.09$; Package-aware $+0.09$; FPP $+0.02$
Easy-tier below-Identity (n=13)	Basic 2; Package-aware 2; FPP 5
Coordinate-probe means	Center/scale 0.311; PCA 0.366; ICP 0.414
ICP probe wins	vs Basic 19/38; vs FPP 24/38
Full + prior − Basic	-0.067 $[-0.113,\ -0.028]$, p=0.002
Full + prior − Basic, valid only	-0.035 (FPP 0.393 vs Basic 0.428)
Full + prior wins vs Basic	13/40 paired wins
Full + prior − Package-aware	-0.059 $[-0.105,\ -0.017]$, p=0.021
Full + prior wins vs Package-aware	15/40 paired wins
Package-aware − Basic	-0.008 $[-0.043,\ +0.029]$, p=0.476
Direct MAE-table tasks	$+0.009$ (n=4, FPP−Basic)
Aggregate-rank-only tasks	-0.079 (n=31, FPP−Basic)
Package-count vs score	r=0.30 (FPP run-level)

E. Observable Trace Excerpts

Table 10. Exact observable trace excerpts for the prototype vignettes in Table 3. Excerpts are copied from agent-visible status messages, command outputs, or runner stderr with whitespace normalized for typesetting. They are not private reasoning transcripts.

Vignette	Source	Exact observable excerpt
Baseline insurance	raw_stdout.jsonl event 27 / agent_message	The expression-guided transform was unstable on one pair, so I'm focusing on rigid ICP variants and selecting among them with an internal cross-slice expression-consistency metric. I hit a runtime failure in one trial...
	raw_stdout.jsonl event 29 / command_output (exit 0)	transform 151673 -¿ 151674 scale 0.9997621288682814 angle_deg 0.015368862857397697 t [-123.35311551 184.1179543] — transform 151675 -¿ 151674 scale 0.9998433740488037 angle_deg 0.0098196030985108 t [-41.59977683 -425.86109631] — transform 151676 -¿ 151674 scale 0.9999418595960391 angle_deg -0.0...
Slow package fallback	raw_stdout.jsonl event 63 / agent_message	I terminated two long-running experimental `paste` jobs to avoid background contention; the validated submission CSVs remain unchanged and ready.
	raw_stdout.jsonl event 61 / command_output (exit -1)	/private/tmp/p2b_spatial_smoke_basic/sa-001-dlpfc-serial-align/.method-venv/lib/python3.11/site-packages/ot/lp/__init__.py:346: UserWarning: numItermax reached before optimality. Try to increase numItermax. — /private/tmp/p2b_spatial_smoke_basic/sa-001-dlpfc-serial-align/.method-venv/lib/python3....
Geometry + expression check	raw_stdout.jsonl event 19 / agent_message	Validation confirms the same transform is best by both geometry and expression consistency, so I'll now write `151673` as anchor coordinates and transform `151674` into that frame, preserving each file's original `obs...
	raw_stdout.jsonl event 15 / command_output (exit 0)	best chamfer 101.33008656260799 A-¿B 51.526502334682114 B-¿A 49.803584227925874 — init 180 1 1 scale 0.9848176923132688
Large multi-slice execution	raw_stdout.jsonl event 21 / agent_message	The serial alignment completed successfully and wrote all 33 CSVs with a large improvement in adjacent-slice consistency (`4740.3` to `12.87` median NN distance). I'm running an independent validation pass on filename...
	raw_stdout.jsonl event 19 / command_output (exit 0)	Loaded 33 slices — Raw adjacent median-NN distance (lower is better): 4740.308 — Aligned adjacent median-NN distance: 12.870 — Wrote 33 CSV files to submission/aligned_coordinates
Package-backed selection	raw_stdout.jsonl event 69 / agent_message	`spateo` successfully produced a usable subset-based transform, and its diagnostic score is slightly better than my custom rigid fallback. I'll now validate output CSV integrity (row order, obs IDs, finite coordinates...
	raw_stdout.jsonl event 83 / command_output (exit 0)	"identity": 0.4121425747871399, — "spateo_affine": 0.529234766960144 — "identity": [— "spateo_affine": [— "selected_by_internal_metric": "spateo_affine",
Package error recovery	raw_stdout.jsonl event 33 / agent_message	The run failed because the two slices have different gene sets. I'm patching the script to intersect/reorder genes first, then rerunning the same `paste`/`paste2` comparison.
	raw_stdout.jsonl event 57 / command_output (exit 0)	"method": "paste_pairwise", — "method": "paste2_partial", — "selected": {
Provider disconnect after work	raw_stdout.jsonl event 62 / agent_message	`PASTE2` remains running on the full pair, so it appears substantially heavier at this scale. I'll give it one final bounded poll; if it still doesn't return, I'll keep the completed `spateo` submission.
	raw_stdout.jsonl event 48 / command_output (exit 0); raw_stderr.txt	———¿ [Models alignment based on morpho, mode: SN-S.] finished [2592.0746s] — wrote submission/aligned_coordinates/mouse1_slice212.csv 6277 — wrote submission/aligned_coordinates/mouse1_slice221.csv 5579 — wrote submission/aligned_coordinates/mouse1_slice232.csv 5169 — wrote submission/aligned_.... — stderr: 2026-05-02T15:17:40.067956Z ERROR codex_api::endpoint::responses_websocket: failed to connect to websocket: IO error: failed to lookup address information: nodename nor servname provided, or not known, url: wss://chatgpt.com/backend-api/codex/responses — 20...

F. Reproducibility Notes

Artifact release. The camera-ready artifact bundle will include the 40 task manifests, sanitized public-workspace builders, scorer and hidden-evaluation protocol, three executable prompt variants, run-structure analyzer, all 360 run artifacts, and bootstrap/paired-test/baseline-probe analysis scripts. Hidden gold coordinates and held-out marker genes ship as scorer-only assets so the benchmark can be rerun end-to-end without exposing gold labels to solve workspaces.

Leakage controls in detail. Every blind solve runs under a read-deny profile that blocks benchmark source assets, scorer implementation files, prior runs, raw-data caches, and analysis tables. Provider API credentials are stripped from the agent environment, and a post-run auditor flags any access matching the deny list. All 360 reported runs are clean; no high-severity findings.

Task identifier note (potential prior). Task identifiers embed dataset-family hints such as DLPFC or MHPR. These names encode standard public SABench dataset families, so the agent can recognize that a task is DLPFC-derived. We treat this as public scientific context, not an unintended leak.

Compute budget and timeouts. Every regime runs under the same hosted coding-agent configuration with high reasoning effort. There is no hard token cap inside the agent loop; the runner's wall-clock timeout is identical across regimes (the prebootstrap step in the Full + prior regime runs *before* the agent loop). We do not normalize wall-clock differences in the headline table because compute behavior is part of the affordance bundle under study.

This draft is intentionally anonymous; acknowledgements and author-identifying repository URLs will be added for the camera-ready version.

G. Prompt and Analysis Templates

The excerpts below document the manuscript-facing prompt structure without exposing local file names, repository paths, or exact run-directory details. All three reported regimes use the same blind solve scaffold and differ in the regime-specific context block: Basic, Package-aware, or Full + prior. The companion artifact bundle preserves the executable prompts verbatim.

Runner command setting. Reported runs used a Codex approval-bypass option to disable approval prompts and sandbox interruptions during autonomous task execution:

```
--dangerously-bypass-approvals-and-sandbox
```

Shared Solve Scaffold (used by all three variants)

```
You are solving a blind spatial-transcriptomics alignment task.
Use only the sanitized task inputs provided in the workspace: expression data, spot
    or cell identities, original spatial coordinates, and slice identities.
No local scorer is available during solving.
Do not search for hidden aligned coordinates, ground-truth transforms, scorer
    labels, prior submissions, benchmark source material, or private caches.
First produce a complete valid aligned-coordinate submission for every required
    slice or pair. Then improve it only with diagnostics that can be computed from
    the allowed task inputs. When finished, report the submitted approach concisely.
Task-specific instructions:
{task instructions}
Regime-specific context:
{Basic, Package-aware, or Full + prior context}
```

Prompt Variant 1: Basic Context

```
No paper-method ranking, package list, or prebuilt method environment is exposed.
Use general numerical and scientific-programming tools available in the solve
```

```
    workspace.
Favor simple, complete, auditable alignments first: identity or centering baselines,
    rigid or affine transforms, expression-neighborhood checks, and per-slice sanity
    checks based only on visible coordinates and expression.
Do not infer hidden labels or transforms from benchmark names. If an attempted
    refinement risks breaking a complete valid submission, keep the safer submission.
```

Prompt Variant 2: Package-Aware Context

```
The task is related to spatial-transcriptomics alignment methods such as
    PASTE/PASTE2, SPACEL, Spateo, STalign, STAligner, scSLAT/SLAT, CAST, GPSA,
    SANTO, and STAIR. These names are method suggestions, not ground-truth labels or
    scorer information.
You may install, import, or reimplement method ideas when practical, but no
    method-specific environment has been prebootstrapped. Before spending time on
    slow installs or large package runs, create a complete valid submission and test
    package calls on a small representative subset.
If a package is unavailable, too slow, or mismatched to the task schema, switch to a
    transparent custom alignment rather than returning no coordinates.
```

Prompt Variant 3: Full + prior context

```
A benchmark paper compared multiple spatial-alignment method families on related
    spatial-transcriptomics tasks. Use these paper-derived priorities as a search
    prior, not as task labels, hidden transforms, or scorer information.
The workspace exposes method-specific Python environments and locally prepared
    package assets. Treat import success as a starting point only: a useful package
    attempt must run on the task data or a representative subset and return usable
    coordinates.
Always create a complete valid submission before slower package attempts, and record
    why each package was kept or abandoned.
Paper-informed priority examples:
- DLPFC/Visium serial alignment: try SPACEL or PASTE2 early; CAST, STAIR, and Spateo
    are additional candidates.
- MERFISH or image-like serial alignment: try Spateo or SPACEL early; PASTE2, STAIR,
    and SLAT are additional candidates.
- NGS/Visium or 3D/MAE stacks: try PASTE2, SPACEL, Spateo, PASTE-family fallbacks,
    STAIR, or SLAT depending on data size and API fit.

Do not treat this list as a hard rule. If slices are large, estimate candidate
    transforms on subsets first. If a method needs absent labels, use only
    unsupervised domains derived from allowed expression or coordinate data. Do not
    invent or recover withheld anatomical labels.
```