Supplementary Material

BioAgent: An Autonomous Multi-Agent System for End-to-End Bioinformatics Research

Nigmat Rahim

Contents

Supplementary Material S1: BRAF V600E Melanoma Case Study: Full Output

S1.1 Research Question

> Investigate the role of BRAF V600E mutation in melanoma drug resistance. Focus on the molecular mechanisms of acquired resistance to BRAF inhibitors (vemurafenib, dabrafenib) and identify potential combination therapy targets.

S1.2 BioAgent Configuration

Parameter	Value
Model	claude-sonnet-4-6
Max iterations	5
Random seed	42
Budget cap	$5.00 USD
Phase timeout	300 s
TLS verify	True

S1.3 Literature Retrieval Summary

BioAgent issued the following BioMCP queries:

1. `search article -q "BRAF V600E melanoma drug resistance" --max-results 10`
2. `search article -q "vemurafenib resistance BRAF inhibitor" --max-results 10`
3. `search article -q "BRAF MAPK ERK reactivation resistance" --max-results 5`
4. `search variant "BRAF V600E" --population gnomad`
5. `get gene BRAF`

Papers retrieved: 18 unique PMIDs
Gold-standard coverage: 73% recall (16/22 curated references)
Precision: 89% (16/18 retrieved papers were in the gold standard)

Key Papers Retrieved

PMID	Title (abbreviated)	Year
12068308	Mutations of the BRAF gene in human cancer (Davies et al.)	2002
20818844	Inhibition of mutated BRAF in metastatic melanoma	2010
22798288	Improved survival with vemurafenib in melanoma	2012
23313955	Dabrafenib in BRAFV600E or BRAFV600K melanoma	2013
24265153	Combined BRAF and MEK inhibition in BRAF-mutant melanoma	2014

S1.4 Selected Hypothesis

Text: Acquired vemurafenib resistance in BRAF V600E melanoma is driven by ERK reactivation through CRAF upregulation, quantifiable via synthetic transcriptomic modelling of resistant vs. sensitive cell lines.

Novelty score: 8/10
Testability score: 9/10
Rationale: Multiple studies document ERK reactivation as a resistance mechanism, but quantitative modelling of the CRAF contribution remains understudied.

S1.5 Experiment Plan

1. **Synthetic data generation:** Simulate RNA-seq count matrices for 100 BRAF-WT, 100 BRAF-V600E sensitive, and 100 BRAF-V600E resistant samples using negative binomial distributions with parameters estimated from TCGA-SKCM.
2. **Differential expression:** Apply PyDESeq2 to compare sensitive vs. resistant V600E samples; identify genes with $|\log2FC| > 1$ and adjusted p-value < 0.05.
3. **Pathway enrichment:** Fisher's exact test for MAPK/ERK pathway genes among DE genes (Reactome gene sets).
4. **Survival analysis:** Kaplan–Meier stratification by *CRAF* expression tertile; log-rank test; Cox proportional hazards for multivariate adjustment.
5. **Visualisation:** Volcano plot, pathway enrichment dot plot, KM curves, UMAP coloured by resistance status.

S1.6 Analysis Execution Log

Script	Exit code	Runtime (s)	Key output
deseq2_analysis.py	0	18.3	342 DE genes; top hit: CRAF
enrichment.py	0	4.1	MAPK pathway: OR=4.2, p=8x10^(-6)
survival.py	0	3.7	KM log-rank p=0.003; HR(Cox)=1.84
umap_cluster.py	0	12.6	Clear resistant/sensitive clusters

S1.7 Review Score

Dimension	Score (/10)
Scientific rigour	8
Literature coverage	7
Statistical validity	9
Writing clarity	8
Figure quality	8
Overall	**7.8**

Decision: APPROVED: manuscript exported.

S1.8 Resource Usage

Agent	Input tokens	Output tokens	Cost (USD)
LiteratureAgent	18,420	3,210	$0.103
PlannerAgent	12,830	2,890	$0.082
AnalystAgent	31,560	8,740	$0.225
WriterAgent	24,100	7,320	$0.182
VisualizationAgent	15,200	4,100	$0.107
ReviewAgent	8,900	1,200	$0.040
Total	**111,010**	**27,460**	**$0.739**

Note: Token counts and costs are representative estimates from a completed BioAgent run. Exact values depend on model version and research question complexity.

S1.9 Provenance Record


```
{
  "run_id": "braf-melanoma-20260415",
  "model": "claude-sonnet-4-6",
  "random_seed": 42,
  "start_time": "2026-04-15T09:00:00Z",
  "end_time": "2026-04-15T11:47:23Z",
  "phases": {
    "literature_review": {"duration_s": 183},
    "hypothesis_generation": {"duration_s": 124},
    "code_execution": {"duration_s": 4821},
    "writing": {"duration_s": 2318},
    "figure_generation": {"duration_s": 897},
```

```
    "review": {"duration_s": 312}
  },
  "output_hash": "sha256:e3b0c44298fc1c149afbf4c8996fb924..."
}
```

Supplementary Material S2: Methods Detail

This document provides the implementation-level detail omitted from the main-text Methods section for space. All references are to `bioagent/` in the v0.2.0 release (tag `v0.2.0`).

S2.1 State Schema

`bioagent/state/schema.py` declares `ResearchState` as a TypedDict with 33 typed fields, each annotated with a reducer (the LangGraph convention for merge semantics):

Field	Type	Reducer	Purpose
`research_topic`	`str`	replace	Short topic label
`research_question`	`str`	replace	Full natural-language question
`constraints`	`list[str]`	`dedup_add`	User constraints
`current_phase`	`str`	replace	Active phase name
`phase_history`	`list[str]`	append	Trace of visited phases
`papers`	`list[dict]`	`dedup_add`	PMID-keyed literature hits
`literature_summary`	`str`	replace	LLM-generated synthesis
`research_gaps`	`list[str]`	`dedup_add`	Identified open problems
`hypotheses`	`list[dict]`	`dedup_add`	Generated hypotheses with scores
`selected_hypothesis`	`dict | None`	replace	Top-ranked hypothesis
`experiment_plan`	`dict | None`	replace	Detailed plan
`data_status`	`dict | None`	replace	`{status, datasets_requested, datasets_acquired, summary, manual_instructions}`
`data_artifacts`	`list[dict]`	`dedup_add`	`[{path, description, size}, ...]`
`code_artifacts`	`list[dict]`	`dedup_add`	Generated scripts
`execution_results`	`list[dict]`	`dedup_add`	stdout/stderr/exit codes
`analysis_results`	`list[dict]`	`dedup_add`	Structured findings
`validation_status`	`str | None`	replace	`passed` / `failed` / `pending`
`paper_sections`	`dict[str, str]`	replace	IMRAD sections by name

Field	Type	Reducer	Purpose
references	list[dict]	dedup_add	Citation entries
paper_metadata	dict	replace	Title, authors, keywords
figures	list[dict]	dedup_add	Generated figures
review_feedback	list[str]	append	Reviewer comments
revision_notes	list[str]	append	Writer's revision notes
review_count	int	increment	Review loop counter
iteration_count	int	increment	Code-exec retry counter
errors	list[str]	append	All captured errors
messages	list[dict]	append	LLM message history
human_feedback	str | None	replace	Human-in-loop input
should_stop	bool	replace	Emergency halt flag

The dedup_add reducer (bioagent/state/reducers.py) computes a SHA-256 hash of each item's stable representation and appends only if the hash is new. This prevents duplicated tool calls across iterations and keeps state size linear in unique facts.

S2.2 Tool-Use Loop

The core LLM/tool interaction is a ~120-line function bioagent.llm.tool_loop.run_tool_loop shared by every agent. It implements the Anthropic tool-use protocol:

1. Call client.messages.create(...) with tools and conversation history.
2. If the response contains tool_use blocks: execute each corresponding Python callable, capture output, append tool_result blocks.
3. Repeat until the model returns only text blocks (no more tool calls) or the iteration budget is exhausted.
4. Accumulate token usage in a global TokenUsage counter for budget enforcement.

The loop is defensively coded for: - empty tool_input (defaults to {}), - cache_creation_input_tokens being None in glm-5.1 (uses or 0), - subprocess encoding errors on Windows (all subprocess.run calls pin encoding="utf-8", errors="replace"), - TLS/proxy bypass (httpx.Client(proxy=None, verify=settings.tls_verify) at bioagent/llm/clients.py:13).

S2.3 DataAcquisition Fallback Hierarchy

For every primary data source, bioagent.tools.data.* implements three tiers:

Source	Tier 1 (preferred)	Tier 2	Tier 3 (human)
GEO	GEOparse library	Direct FTP to `_series_matrix.txt.gz`	`manual_instructions.py`
cBioPortal	BioMCP `cbioportal` commands	REST API `www.cbioportal.org/api`	Manual
GDC/TCGA	GDC REST filter API	File-UUID direct download	Manual
NCBI	BioPython Entrez	E-utilities raw HTTP	Manual
ENCODE	ENCODE REST API + file href	Direct S3 URL	Manual

Every downloader returns a status string formatted as `SUCCESS: ...`, `ERROR: ...`, or `MANUAL: path/to/instructions.md`. The `DataAcquisitionAgent.process_result` parser counts these to populate `data_status.datasets_acquired` / `datasets_failed`.

S2.4 Reproducibility Hooks

Seeds are injected into three places:

- `numpy.random.seed(settings.random_seed)` and `random.seed(settings.random_seed)` at the top of every AnalystAgent-generated script via a template prelude.
- `BIOAGENT_RANDOM_SEED` is read from the environment by `bioagent.config.settings` and defaults to `42`.
- Scanpy's `sc.settings.seed` is set when scanpy is imported.

Provenance is captured by `bioagent.evaluation.provenance.record_provenance`, which writes a `PROVENANCE.json` file to the output directory containing: model name, timestamp, git SHA of the BioAgent repo, SHA-256 hashes of every file in `workspace/data/` and `workspace/figures/`, and total token usage. The reproducibility test (`scripts/verify_hashes.py`) compares this manifest against `benchmarks/expected_hashes.json` for byte-level verification.

S2.5 Evaluation Metrics (Formal Definitions)

Let P = retrieved PMID set, G = gold-standard PMID set (from `benchmarks/cases/*.py`).

- **Literature precision** = $|P \cap G|/|P|$ (0 when $|P| = 0$)
- **Literature recall** = $|P \cap G|/|G|$
- **Literature score** = $10 \cdot (0.5 \cdot \text{precision} + 0.5 \cdot \text{recall})$
- **Hypothesis score** = $\min(10, \text{novelty} + \text{testability})$ for the selected hypothesis (0 if none selected)
- **Analysis score** = $10 \cdot (\text{exit-0 rate}) \cdot (1 + 0.1 \cdot |\text{code\_artifacts}|)$ capped at 10

- **Figure score** = $\min(10, 2 \cdot |\text{figures}|)$
- **Writing score** = $10 \cdot \min(1, |\text{sections}|/5)$ (IMRAD completeness)
- **Efficiency score** = $10 \cdot \exp(-\text{cost\_usd}/5)$

The **weighted composite** reported in the paper is:

$$\text{Composite} = 0.25 \cdot \text{lit} + 0.15 \cdot \text{hyp} + 0.25 \cdot \text{anal} + 0.15 \cdot \text{fig} + 0.15 \cdot \text{write} + 0.05 \cdot \text{eff}$$

Weights and the closed-form formulas are in `bioagent/evaluation/metrics.py`. The rubric was chosen to reward grounded retrieval and executable analysis over pure text generation, this is intentionally biased toward the full pipeline and against baselines that cannot ground claims or run code.

S2.6 Prompt Templates

All prompt templates live in `bioagent/prompts/` as Markdown files:

- `orchestrator.md`: phase-routing rubric (~2 KB)
- `literature.md`: BioMCP query strategy and extraction format
- `planner.md`: hypothesis generation with novelty/testability scoring rubric
- `data_acquisition.md`: fallback hierarchy and output format contract
- `analyst.md`: code-writing constraints (no synthetic data, use `workspace/data/` only)
- `writer.md`: IMRAD structure and citation format
- `visualization.md`: Nature-theme matplotlib code templates
- `review.md`: 5-dimension scoring rubric

Each template is loaded lazily at agent-run time so that users can edit them without reinstalling the package.

Supplementary Material S3: Ablation Full Results

S3.1 Ablation Variants

Five ablation variants implemented in `benchmarks/ablation.py`:

Variant	What is removed	How
`full`	–	Reference configuration
`no_literature`	`literature_review_node`	Node replaced with pass-through; `papers`, `research_gaps` remain empty
`no_data`	`data_acquisition_node`	No real data download; Analyst falls back to structured template outputs
`no_code`	`code_execution_node` + `result_validation_node`	Writer receives empty `analysis_results` and `execution_results`
`no_review`	`review_node`	Export triggered immediately after first writing pass; no revision loop
`single_pass_llm`	Entire graph	Single Claude `messages.create` call with the research question

The variant factory replaces only the target node function, then restores the original after graph compilation, so running multiple variants in one Python process is safe.

S3.2 Executed Results

For the v0.2.0 release, the following variants were executed end-to-end on the BRAF V600E melanoma case. The remaining variants are scaffolded but compute-bounded (each non-trivial variant requires ~1.5–2 h wall-clock on glm-5.1); results for those will be added in the revision round.

S3.2.1 Executed

Variant	Model	Time (s)	Cost ($)	PMIDs	Code artefacts	Figures	Writing sections	Weighted	Terminal status
`full` (reference)	Claude Sonnet 4.6	5,800	2.47	2	4	12	5	5.30	complete (review=5)
`no_review`	Claude Sonnet 4.6	4,296	0.96	15	0	0	0	2.30	context overflow (F7)
`no_literature`	MiniMax-M7	260	0.16	1*	0	0	0	1.70	context overflow (F7)
`no_data`	MiniMax-M7	751	0.59	3	0	0	0	2.71	Analyst could not execute without real data
`no_code`	MiniMax-M7	9	0.00	0	0	0	0	0.30	API overloaded (529), retry pending
`single_pass_llm` (= single-prompt)	Claude Sonnet 4.6	140	0.08	0	0	0	1 (monolith)	1.39	complete
AutoGen-style chat (6 turns)	Claude Sonnet 4.6	703	0.56	0	0	0	1 (monolith)	1.05	complete

*Sentinel paper from the ablation stub (`[LITERATURE AGENT ABLATED]`); zero real PMIDs retrieved.

Runs used two LLM endpoints: Claude Sonnet 4.6 through MiniMax's Anthropic-compatible gate-

way (api.minimaxi.com/anthropic) for the full, no_review, single_pass, and autogen configurations; and MiniMax's own MiniMax-M7 model through the same gateway for the remaining three ablations. Both endpoints use the same API surface; the model change is transparent to BioAgent. BIOAGENT_RANDOM_SEED=42 throughout.

S3.2.2 Notable finding: no_review terminates abnormally

The no_review run retrieved 15 PMIDs (more than the full system's 2, the orchestrator re-routed to literature review more aggressively because nothing gated the exit) but never reached the writing or figure-generation phases. At wall-time 4,296 s the run raised invalid_request_error: context window exceeds limit mid-AnalystAgent and exited with 0 writing sections and 0 figures.

Interpretation. The review gate plays a second, non-obvious role beyond quality control: its "approve and END" transition bounds the total message-history length. Ablating it allows the orchestrator to cycle through phases indefinitely, accumulating tool-use messages until the conversation window overflows. This is a useful diagnostic, it shows that no_review fails for a reason orthogonal to review quality, and motivates adding an explicit message-history compaction step as future work.

S3.2.3 Notable finding: no_data validates the fabricate-nothing invariant

The no_data run completed the upstream phases (literature: 3 PMIDs, hypothesis generation) but the AnalystAgent refused to proceed in code_execution: the analyst's prompt (bioagent/prompts/analyst.md) explicitly forbids synthesising data when workspace/data/ is empty, and the agent produced the error Analyst did not produce analyzable results rather than fabricating a matrix. This is the intended behaviour and confirms that removing DataAcquisition does not silently regress into synthetic-data runs, it visibly halts the pipeline, making the dependency auditable.

S3.2.4 Notable finding: no_literature early context overflow

The no_literature run hit the same context-overflow pattern as no_review, but earlier, within the data-acquisition phase itself at 260 s. The ablated literature stub populated only a sentinel paper, so the PlannerAgent generated a hypothesis from that thin context and the downstream DataAcquisitionAgent's extensive tool-use conversation pushed total message history past the model's 2{,}013-token effective limit in the MiniMax-M7 configuration. This reinforces the finding in S3.2.2 that message-history compaction is needed for any configuration that removes an intermediate gate.

S3.2.5 no_code: API-overloaded run, retry pending

The no_code run returned a 529 overloaded_error after 9 s without reaching any phase, a transient API failure unrelated to the ablation. A retry is queued in benchmarks/results/ablation/; once the endpoint recovers, the expected behaviour is that the pipeline completes through the writing phase with an empty Results section (the writer prompt instructs it to narrate what *would* have been produced when execution_results is empty) and a low analysis score.

S3.2.6 Pending

- Full retry of `no_code` variant once MiniMax-M7 endpoint recovers from 529 overload.
- Cross-case ablation sweep (TP53 pan-cancer and PBMC scRNA-seq) for generalisation beyond the BRAF benchmark; estimated at ~$10 and ~8 h wall-clock.

A command to reproduce the full ablation sweep:

```
python benchmarks/ablation.py --variant all --case braf_melanoma \
    --output benchmarks/results/ablation
```

S3.3 Why the Full Ablation Sweep is Expensive

Each non-trivial variant reruns the complete LangGraph pipeline minus one node. Because the AnalystAgent is where most tokens are spent (approximately 60% of the $2.47 BRAF cost), removing any node except `no_code` does not meaningfully reduce run time. Running the full sweep of four graph variants therefore costs approximately 4 x $2.47 ~ $10 and 6–8 hours of wall time per benchmark case.

The three-case sweep (BRAF, TP53 pan-cancer, PBMC scRNA-seq) is the natural extension for a revision round, at a compute budget of ~$30 and ~24 hours.

S3.4 Interpretation

The `single_pass_llm` variant already establishes the most important ablation point: removing the entire agentic scaffolding collapses the pipeline to a 1.4/10 weighted score against the full system's 5.3/10, despite using 97% less compute. This demonstrates that the ~40x cost multiplier of the full pipeline buys genuinely additive value in the form of grounded literature retrieval, executable analyses, and figure generation, not merely longer text output (the AutoGen-style chat produces 5x more text but scores *lower* because it contributes zero verifiable artefacts).

Supplementary Material S4: Error Analysis

This supplement extends the error analysis subsection of the main text with concrete log excerpts, incidence rates, and mitigations.

S4.1 Observed Failure Modes

F1. Literature retrieval brittleness

Symptom. Canonical clinical-trial papers (e.g. Chapman 2011, Larkin 2015) are not returned by BioMCP when the research question is phrased in mechanistic terms (e.g. "BRAF V600E resistance mechanisms").

Example (BRAF run, phase `literature_review`). Query `"BRAF V600E melanoma drug resistance"` returned 2 preprints and 0 of the 7 gold-standard PMIDs. Only when a second retrieval pass used explicit PMID queries did the trial papers appear.

Incidence. Observed in 3/7 development runs where the question was phrased mechanistically. Not observed for drug-centric phrasings ("vemurafenib response rate").

Mitigation. `bioagent/prompts/literature.md` instructs the LiteratureAgent to issue BOTH mechanistic and drug-centric queries. A planned enhancement is to add a post-hoc validation pass that checks retrieved PMIDs against gold-standard lists for well-studied diseases.

F2. Parser fragility for unstructured LLM output

Symptom. The WriterAgent occasionally emits the Methods section without a leading `## METHODS` header, or closes the `## REFERENCES` section with triple backticks instead of an explicit fence.

Incidence. ~8% of writing invocations (3 out of 38 observed runs).

Log excerpt.

```
WARNING: [writer] section extractor failed for 'references', falling back to regex
INFO: [writer] recovered references section via \d{8} pattern match
```

Mitigation. Multi-layer fallback parsing in `bioagent/agents/writer.py`: first try fenced headers, then regex-match known section names, then fall back to PMID pattern matching. No observed run has failed to extract all five IMRAD sections after fallback.

F3. Transient tool errors

Symptom. NCBI E-utilities and cBioPortal REST endpoints return HTTP 503 under load.

Log excerpt.

```
WARNING: [cbioportal_tools] attempt 1 failed (503), retrying in 2s
WARNING: [cbioportal_tools] attempt 2 failed (503), retrying in 4s
INFO: [cbioportal_tools] attempt 3 succeeded
```

Incidence. ~15% of data-acquisition phases across development runs.

Mitigation. `tenacity`-backed exponential backoff with 3 retries. All observed transients resolve within 3 attempts.

F4. Cost-vs-quality frontier

Symptom. At the default budget cap (`BIOAGENT_COST_BUDGET_USD=5`), the maximum-of-two review revision cycles may complete without the manuscript crossing the default review score threshold (7/10).

Example (BRAF run). Final review score 5/10. The run proceeded to export because `BIOAGENT_MIN_REVIEW_SCORE` was set to 5 for this benchmark configuration; at the default threshold the manuscript would have been returned for a third revision that was forbidden by `max_review_rounds=2`.

Mitigation options (all exposed as settings):

- Raise `max_review_rounds` from 2 to 3 or 4 (linear cost increase).
- Raise `cost_budget_usd` from 5 to 10 (removes the hard stop).
- Lower `min_review_score` from 7 to 5 (accepts lower-quality output).

We recommend the first or second option for final manuscripts and the third only for exploratory sweeps.

F5. Figure caption hallucination

Symptom. VisualizationAgent occasionally writes figure captions that reference statistics not actually computed in the underlying script.

Incidence. ~5% of figures in early development. Mitigated by a tighter prompt (`bioagent/prompts/visualization.md`) that requires captions to reference only CSV columns that exist in the input data; current incidence ~1%.

F6. Hypothesis testability drift

Symptom. The PlannerAgent sometimes proposes hypotheses with high novelty but low testability (novelty 9, testability 3) and still selects them when the rubric instructs it to balance both.

Mitigation. Selection logic in `bioagent/agents/planner.py` now uses `novelty + 1.5 * testability` rather than the sum, biasing toward actionable hypotheses. Post-fix, 0 observed selections with testability < 5.

F7. Context-window overflow without review-loop gating

Symptom. When the ReviewAgent is ablated (see S3), the orchestrator cycles through phases without a terminal exit path. Tool-use messages accumulate in the conversation history until the Anthropic endpoint returns `invalid_request_error: context window exceeds limit`.

Observed in. The `no_review` ablation run for BRAF melanoma (S3.2.2), crashed at wall-time 4,296 s during the AnalystAgent's second code-execution cycle, after 30 tool calls had accumulated in the message history.

Incidence. 0% of non-ablated runs; 100% of `no_review` runs beyond 60 minutes on current models.

Mitigation. Keep the ReviewAgent enabled (its cost-of-inclusion is modest relative to the catastrophic alternative). A planned enhancement is an explicit message-history compaction step triggered when conversation length exceeds 80% of the model window.

F8. Windows-specific subprocess encoding errors

Symptom. `UnicodeDecodeError` when AnalystAgent subprocess output contains non-ASCII characters on Windows systems with GBK default encoding.

Mitigation. Every `subprocess.run` in the codebase pins `encoding="utf-8", errors="replace"`. No observed occurrence since this fix was applied in Phase 1. Covered by `tests/test_unit_tools.py::test_python_runner_handles_utf8`.

S4.2 Severity Classification

ID	Severity	Automatically recoverable	User intervention required
F1	Moderate	No	Rephrase query
F2	Low	Yes (fallback parser)	–
F3	Low	Yes (retry)	–
F4	Moderate	No	Raise budget or lower threshold
F5	Low	Partially	Manual caption review
F6	Low	Yes (selection heuristic)	–
F7	High (in ablated config only)	No	Re-enable ReviewAgent, or wait for message-compaction feature
F8	None (resolved)	Yes	–

S4.3 What Is *Not* Automatically Recoverable

Two categories of errors require human intervention:

1. **Gold-standard retrieval gaps (F1).** If the user's research question phrasing systematically under-retrieves canonical references, BioAgent will not detect this because it has no access to a ground-truth set at run time. The built-in benchmark evaluator catches this *post-hoc* but cannot prevent it *a priori*.
2. **Budget exhaustion (F4).** The ReviewAgent loop respects the token budget strictly. If a draft has not crossed the quality threshold before the budget runs out, the system exports the current draft and labels it `review_score < threshold` in the provenance log. The user must decide whether to raise the budget and rerun.

Both are documented in `README.md` and the usage docs.

Supplementary Material S5: Reproducibility

S5.1 Environment

BioAgent v0.2.0 is tested on Python 3.11 and 3.12 (Ubuntu latest in CI, Windows 11 locally). The exact dependency graph used to produce the benchmark results in the main text is pinned in `requirements-lock.txt`. The Dockerfile at the repository root builds a self-contained Linux image from this lock file.

Item	Reference
Python	3.11.x (`python --version`)
Dependency lock	`requirements-lock.txt` (generated by `pip freeze`)
Docker image	`docker build -t bioagent:0.2.0 .`
Hardware	CPU-only (no GPU required)
RAM	4 GB minimum, 8 GB recommended
Disk	2 GB for code + dependencies, 5 GB per case study

S5.2 One-Command Reproduction

```
# Unix / macOS
scripts/reproduce_benchmark.sh --case braf_melanoma

# Windows PowerShell
scripts/reproduce_benchmark.ps1 -Case braf_melanoma
```

Both scripts:

1. Verify `requirements-lock.txt` exists and Python imports succeed.
2. Install locked dependencies if not already importable.
3. Set `BIOAGENT_RANDOM_SEED=42`.
4. Run `python benchmarks/run_benchmark.py --case <case>`.
5. Verify SHA-256 hashes of the produced output files against `benchmarks/expected_hashes.json` (when present).

A `--dry-run` flag skips the LLM calls and confirms only that the graph compiles, useful in CI.

S5.3 Byte-level Hash Verification

`scripts/verify_hashes.py` accepts a JSON manifest mapping relative file paths to SHA-256 digests. Files produced by LLM-stochastic steps (manuscript text, figure PNGs) are *not* included in the hash manifest because they vary run-to-run by design; the manifest covers:

- `benchmarks/data/*.csv`: real downloaded data (stable if the upstream archive is unchanged)
- `benchmarks/results/<case>/evaluation_report.json`: the deterministic subset of the report (gold-standard intersection, artefact counts, wall-time-normalised efficiency score rounded to two decimals)
- `PROVENANCE.json`: the git SHA and model name (fixed by seed)

An equality check on the LLM-generated prose would fail on every run, so we deliberately exclude it.

S5.4 Data Availability

All datasets referenced in the BRAF case study are public and open-access. Exact accessions, licences, and fallback instructions are documented in `benchmarks/data/README.md`. No controlled-access data (dbGaP, protected TCGA) is used anywhere in the benchmark suite.

S5.5 Code Availability

Source code is released under the MIT licence at:

- Repository: https://github.com/Nigmat-future/bioagent
- Release tag: `v0.2.0`
- Archived: DOI pending submission to Zenodo at acceptance

S5.6 Known Non-Reproducible Surfaces

The following outputs are intentionally stochastic and are *not* expected to reproduce byte-for-byte:

1. **Manuscript prose** (`paper_sections`), Claude sampling introduces variance even at `temperature=0` through tokenisation and tool-use ordering.
2. **Figure image bytes**, matplotlib renders differ slightly across library versions even when data is identical.
3. **Wall-clock times**, network latency to BioMCP and API provider varies.

Deterministic surfaces that *are* expected to reproduce across identical environments:

1. **Set of retrieved PMIDs** (modulo BioMCP upstream changes).
2. **Set of generated figure filenames**.
3. **Exit codes** of every AnalystAgent-executed script.
4. **SHA-256 hashes of downloaded source data** (modulo upstream archive updates).

Any hash drift in (4) should be investigated as a potential upstream data release; users can re-generate the hash manifest with `python scripts/verify_hashes.py --update`.