ProCARE: Real-World Study Automation via Profile-Grounded Evidence Contracts

Zhaoyu Fan[1,3*] **Bowen Han**[1,3*] **Jincheng Ou**[1] **Yanwei Ren**[1,2] **Zehua He**[1] **Kaiyu Huang**[1]
Pengcheng He[1] **Shanshan Ni**[1] **Chengchen Gong**[1] **Shuai Xue**[3] **Qingjiang Shi**[1,4†]
[1]Tongji University
[2]University of Electronic Science and Technology of China
[3]Shanghai Hengfang Health Technology Co., Ltd.
[4]Shenzhen Research Institute of Big Data

Abstract

Real-world studies (RWS) analyze routinely collected healthcare data, such as electronic health records and insurance claims, to estimate treatment effects, monitor safety, and identify risk factors outside randomized trials. Automating RWS with large language model agents is appealing but brittle: valid analyses depend on dataset-specific clinical semantics, including index dates, follow-up windows, patient/event-level units, and outcome boundaries that are often implicit in observational data. This leads to schema mismatches, temporal errors, inconsistent analytical granularity, and unsupported clinical claims. We present ProCARE, a profile-grounded framework that reformulates RWS automation as constrained compilation. ProCARE converts heterogeneous data profiles into Evidence Contracts specifying available variables, temporal attributes, distributions, and evidentiary limits, and compiles study objectives into RWS-IR, a strongly typed intermediate representation for variables, time relations, analysis units, and statistical tasks. Deterministic validation and localized repair then check and revise generated workflows across schema mapping, cohort construction, variable derivation, analysis, and interpretation. On 36 EHR-based benchmark tasks, ProCARE outperforms strong LLM-agent baselines in code executability, semantic consistency, and conclusion traceability, with automated violation diagnoses showing moderate-to-substantial agreement with blinded expert review.

1 Introduction

Real-world studies (RWS) derive clinical evidence from observational healthcare data such as electronic health records and insurance claims [1]. Unlike generic table analysis, RWS workflows are constrained by relational structure, clinical variable semantics, temporal anchors, analysis grain, and evidence scope. Cohorts must be assembled across heterogeneous tables, variables must be assigned valid clinical roles, exposure and follow-up must be anchored consistently, and reported findings must remain within the support of the executed analysis. These requirements make RWS automation a structured scientific generation problem rather than a plain code-generation task.

LLM-based agents have been applied to data science and scientific workflow automation [2–4], but their standard interfaces do not preserve all constraints required by schema-rich clinical data. Textualizing multi-table datasets can obscure entity keys, visit structure, table linkage, missingness, and time-dependent variables [5–7]. RWS requires domain-specific reasoning about endpoint definitions,

*Equal contribution.
†Corresponding author. Email: shiqj@tongji.edu.cn.
GitHub repository: https://github.com/ZyFanTJ/ProCARE.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

censoring, confounding, proxy variables, and claim scope [8, 9]. We observe three recurring failure modes: *query-to-specification failures*, where study concepts are bound to missing, incompatible, or ambiguous fields; *specification-to-code failures*, where free-form code violates the declared join graph, analysis grain, temporal ordering, or artifact protocol; and *artifact-to-claim failures*, where descriptive statistics or proxy artifacts are converted into unsupported clinical claims. These failures indicate missing contracts between the data, study specification, executable workflow, and final report—a structural bottleneck that bounds even strong baselines like Claude Code.

ProCARE addresses this gap by formulating RWS automation as *contract-valid generation*. Its object is a heterogeneous data profile Φ, which records schema structure, field statistics, clinical semantics, temporal roles, validity flags, and assumptions. Rather than serving as a context summary, Φ acts as a profile-grounded evidence contract. ProCARE compiles each user goal into a typed RWS intermediate representation Z, turns Z into executable artifacts through a profile-constrained coding harness, validates execution with deterministic static and runtime gates that produce localized violation sets $\mathcal{U}_t$, and admits report claims only through evidence certificates e_q that link claims to validated artifacts, executed code, profile entries, or recorded assumptions [10]. The output is an auditable artifact bundle containing the profile, RWS-IR, generated code, execution logs, result and figure manifests, validation evidence, assumptions, evidence certificates, and final report.

Contributions. (i) a heterogeneous RWS profile and typed RWS-IR that define a study contract over data structure, clinical semantics, temporal roles, analysis grain, and assumptions; (ii) evidence certificates that restrict factual report claims to validated artifacts, executed code, profile entries, IR fields, or assumptions; and (iii) an end-to-end RWS evaluation over 36 HCC tasks and 18 public-health transfer tasks, measuring executability, code quality, artifact completeness, and claim traceability.

2 Related Work

LLM agents for data analysis. Recent LLM agents combine reasoning, tool use, code execution, and feedback for multi-step data-analysis and scientific workflows [11–14]. Data-analysis systems and benchmarks such as InfiAgent-DABench, Data Interpreter, DSBench, and DataSciBench evaluate agents that translate natural-language goals into programs, tables, visualizations, or model outputs [15–18]. Despite this progress, these systems still report failures in state tracking, dependency management, numerical consistency, and data-dependent repair. In RWS, such errors become methodological failures because executable code must preserve relational structure, analysis grain, temporal anchors, clinical roles, and claim scope. ProCARE addresses this setting with contract-valid generation over typed study objects, executable artifacts, and certified claims.

Structured and clinical-data grounding. Another line of work improves LLM grounding over spreadsheets, tables, SQL databases, and structured clinical records. Table-centric methods compress, encode, or retrieve from large tabular structures [19–23], while text-to-SQL, structured-EHR, and FHIR benchmarks evaluate reasoning over relational or resource-based clinical data [24–26]. These methods improve structured-data access, but they mainly target retrieval, query answering, or task-level execution. RWS automation further requires variable bindings, joins, temporal anchors, artifacts, and claims to remain valid under one study-level contract. ProCARE enforces this contract by propagating a heterogeneous profile through goal-to-IR compilation, profile-constrained execution, and evidence-certified reporting.

RWS evidence constraints. RWE generation is governed by methodological, reporting, and regulatory constraints beyond table access or code execution. STROBE and RECORD emphasize transparent reporting of study design, data sources, variables, bias, missingness, and provenance [27, 28]; target-trial emulation and TARGET guidance require explicit protocols, estimands, eligibility criteria, treatment strategies, follow-up, assumptions, and observational-data mappings [29, 30]. OMOP, FHIR, and FDA guidance further emphasize standardized representation and reliable provenance for RWD use [31–33]. Recent RWE automation benchmarks test whether LLM agents can conduct observational analyses or reproducible hypothesis testing from medical databases [34, 35]. ProCARE incorporates these constraints into generation by compiling typed RWS-IR, localizing execution failures as contract violations, and admitting factual claims only through evidence certificates.

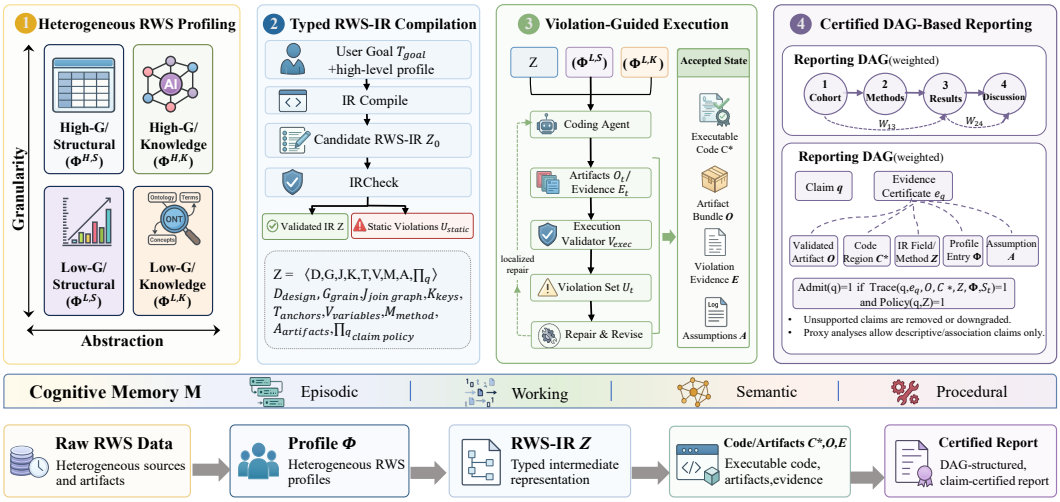


Figure 1: ProCARE architecture for contract-valid RWS automation. A heterogeneous RWS profile constrains RWS-IR compilation, violation-guided execution, and evidence-certified DAG reporting as a profile-grounded evidence contract.

3 Method

ProCARE casts RWS automation as contract-valid generation over the three profile-constrained transitions in Figure 1. Given a heterogeneous profile Φ and user goal T_{goal}, it produces a typed study specification, validated code, result artifacts, a report, and admissible claims:

$$\Phi, T_{\text{goal}} \xrightarrow{\text{profile contract}} (Z, C^*, O, R, Q), \tag{1}$$

where Z is the RWS-IR, C^* validated code, O result and figure artifacts, R the report, and Q admitted claims. The contract requires every study choice, code operation, and claim to trace to profile entries, validated artifacts, or explicit assumptions.

ProCARE enforces this contract at three boundaries. Query-to-IR compilation admits only profile-supported study designs, variables, temporal anchors, and methods. IR-to-code execution accepts only runs that preserve the declared analysis grain, join graph, temporal ordering, and artifact requirements. Artifact-to-claim reporting admits only claims with evidence certificates linked to validated artifacts, executed code, IR fields, profile entries, or recorded assumptions.

The generation state is maintained as typed artifacts $\mathcal{S}_t = \{\Phi, Z_t, C_t, O_t, \mathcal{E}_t, \mathcal{U}_t, \mathcal{A}_t\}$. Here $\mathcal{E}_t$ stores validation evidence, $\mathcal{U}_t$ contract violations, and $\mathcal{A}_t$ assumptions and limitations.

3.1 Heterogeneous Data Profiling

ProCARE first converts raw RWS inputs into a heterogeneous profile $\Phi = (\Phi^{\text{H,S}}, \Phi^{\text{H,K}}, \Phi^{\text{L,S}}, \Phi^{\text{L,K}})$, indexed by profile level (dataset/task vs. field/instance) and abstraction (structural vs. knowledge-driven). Table 1 summarizes the four blocks. Structural blocks specify joins, grain, temporal anchors, and artifact constraints; knowledge blocks record clinical roles, ontology mappings, proxy status, assumptions, and claim scope. Deterministic parsers extract structural profiles, while ontology checks over UMLS and SNOMED CT validate candidate clinical mappings for semantic type, units, ranges, and temporal-role compatibility. Failed or ambiguous mappings are marked as semantically unknown in $\Phi^{\text{L,K}}$ with their failure causes.

Semantically unknown fields may support only structure-grounded operations: they cannot bind clinical roles, derive endpoints, or support claims beyond their validated level unless a proxy or explicit assumption is recorded. Thus, survival, safety, and utilization queries are compiled from validated entity levels, linkage paths, temporal anchors, clinical roles, and support levels. Unsupported choices are filtered, downgraded, or reported as limitations before IR compilation.

Table 1: Heterogeneous data profile Φ organized by profile level and abstraction.

	High-level profile *(dataset/task)*		**Low-level profile** *(field/instance)*	
Structural	$\Phi^{\mathrm{H,S}}$	Clinical form registry; subject/visit/table linkage; cohort-level missingness.	$\Phi^{\mathrm{L,S}}$	Field type, unit, range, date fields, temporal anchors, and window-specific missingness.
Knowledge	$\Phi^{\mathrm{H,K}}$	Cohort and index-date schema; exposure, outcome, censoring, confounder, and claim-scope roles.	$\Phi^{\mathrm{L,K}}$	Ontology mapping; unit/value validation; temporal-role validation; field-level proxy and confounder roles.

3.2 RWS-IR Compilation and Violation-Guided Execution

The IR and execution stages implement the first two contract transitions: query-to-IR and IR-to-code. The compiler turns the user goal into a typed RWS-IR, while the execution stage tests whether the generated analysis can be run without violating the profile, the IR, or the artifact policy.

RWS-IR compilation. Given a research goal T_{goal}, ProCARE first generates a candidate RWS-IR using the high-level profile:

$$Z_0 = f_{\mathrm{IR}}\left(T_{\mathrm{goal}}, \Phi^{\mathrm{H,S}} \oplus \Phi^{\mathrm{H,K}}\right), \tag{2}$$

where $\oplus$ denotes structured fusion of profile blocks. The candidate IR is admitted only after static profile checking:

$$(Z, \mathcal{U}_{\mathrm{static}}) = \mathrm{IRCheck}_{\mathrm{static}}(Z_0, \Phi, \mathcal{S}_t). \tag{3}$$

The accepted IR is $Z = \langle D, G, J, K, T, V, M, A, \Pi_q \rangle$, where D, G, J, K, T, V, M, A, and Π_q denote study design, analysis grain, join graph, entity keys, temporal anchors, variable-role bindings, method specification, required artifacts, and claim policy, respectively.

Static checks enforce three conditions. First, all referenced tables, fields, and keys must appear in $\Phi^{\mathrm{L,S}}$. Second, clinical roles such as exposure, outcome, censoring, and confounder must be supported by $\Phi^{\mathrm{L,K}}$, marked as validated proxies, or explicitly recorded as assumptions. Third, the declared join graph, index date, follow-up window, and analysis grain must be compatible with the profiled schema. Only IRs with $\mathcal{U}_{\mathrm{static}} = \emptyset$ enter code generation. Otherwise, ProCARE repairs the IR, downgrades the unsupported request, or emits an infeasibility statement. Row-level temporal and grain checks are deferred to execution-time validation.

Profile-constrained execution. Given an admitted IR, ProCARE generates executable analysis code inside a workspace containing Z, $\Phi^{\mathrm{L,S}}$, $\Phi^{\mathrm{L,K}}$, the claim policy Π_q, allowed tools, coding policies, artifact validators, and execution logs. A run is accepted only if it produces validated code C^*, complete output artifacts O, validation evidence $\mathcal{E}$, and disclosed assumptions or limitations $\mathcal{A}$.

Execution is controlled by deterministic gates. Schema gates require every table, field, join key, and output column to be traceable to $\Phi^{\mathrm{L,S}}$. Method gates require cohort filters, index dates, exposure definitions, endpoint or censoring rules, covariates, temporal windows, and statistical choices to match Z and the validated semantic roles in $\Phi^{\mathrm{L,K}}$. Artifact gates require the selected entry point, such as `analysis.py` or `main.py`, to run successfully, avoid failure sentinels, and produce complete result and figure manifests. After each execution attempt, validators return a violation set $\mathcal{U}_t = V_{\mathrm{exec}}(Z_t, C_t, O_t, \mathcal{E}_t, \Phi, \mathcal{S}_t)$. Each violation records the failed contract clause, offending code or artifact, nearest valid profile entries, and permitted repair action. When $\mathcal{U}_t \neq \emptyset$, the next edit is conditioned on the localized failures, $(Z_{t+1}, C_{t+1}) = A_\theta(Z_t, C_t, \mathcal{U}_t, \mathcal{S}_t)$. The violation set guides repair but does not imply global edit minimality.

If a requested endpoint, exposure, covariate, time window, or statistical analysis is unsupported by the profile, ProCARE terminates with a transparent failure trace or downgrades the task to a supported descriptive analysis with explicit limitations. Failed traces remain in $\mathcal{S}_t$ for inspection or subsequent runs. This makes profile constraints and methodological assumptions observable during code generation, rather than implicit in prompt context.

3.3 Certified DAG-Based Reporting

The reporting stage enforces the artifact-to-claim transition: only factual claims supported by validated evidence and allowed by the claim policy are admitted. For each claim q, the report generator emits an evidence certificate $e_q = \langle o_q, c_q, z_q, \phi_q, a_q \rangle$, binding the claim to a supporting artifact, executed code region, IR field or method, profile entry, and optional assumption or limitation. A claim is

admitted only if both traceability and policy checks pass:

$$\mathrm{Admit}(q) = \mathbb{I}[\mathrm{Trace}(q, e_q, O, C^*, Z, \Phi, \mathcal{S}_t) = 1 \wedge \mathrm{Policy}(q, Z, \Pi_q) = 1]\,. \tag{4}$$

The trace condition grounds q in validated artifacts, executed code, IR fields, profile entries, or disclosed assumptions; the policy condition checks whether its type and wording are allowed by $Z.\Pi_q$ at the available support level. Unsupported claims are removed or rewritten as limitations. Proxy-based or descriptive analyses may support descriptive or association claims, but not causal, safety-critical, or high-stakes claims without the required design assumptions and validations.

After claim filtering, a dependency DAG orders report sections: nodes are report components, and edges encode prerequisites such as cohort definition, variable construction, temporal anchors, sample support, and method-design consistency. The DAG controls section order, while evidence certificates control claim admissibility. The support score and traversal rule are provided in the appendix.

4 Evaluation and Benchmark

The benchmark evaluates complete RWS artifact bundles rather than isolated code outputs, covering study specification, code synthesis, visualization, and scientific reporting. Because ProCARE natively emits typed IRs, verifier traces, evidence certificates, and report-structure metadata, we distinguish artifact-level comparative metrics from contract-native diagnostics. Deterministic validators assess checkable constraints such as executability, schema use, temporal/grain consistency, artifact completeness, and claim support. Rubric-based judges are used only for residual semantic ambiguity, presentation quality, or stylistic criteria.

Task construction. The primary benchmark contains 36 instances from four RWS domains: *Cohort*, *Safety*, *Efficacy*, and *Economics*. Each domain includes three research questions, and each question is instantiated under three prompt-specificity levels: GOAL, GOAL+APPROACH, and GOAL+METHOD. All instances use the same longitudinal HCC cohort described below. Agents receive raw relational tables and natural-language prompts under a unified protocol.

Output contract. Each prompt contains **Task**, optional **Guidance**, and **Deliverables**. The expected bundle includes: (i) a study specification, or equivalent intermediate description, covering tables, keys, variables, temporal anchors, methods, artifacts, and claim policy; (ii) an executable entry point, such as `analysis.py` or `main.py`, with artifacts; and (iii) a structured report with assumptions, limitations, and evidence links for factual claims. ProCARE instantiates this contract with typed RWS-IRs and evidence certificates; non-ProCARE systems are evaluated via submitted or post-hoc parsed specifications, manifests, and report-level evidence links. Hallucinated tables or columns are disallowed. GOAL+METHOD must follow specified variables and terminate when required inputs are missing; GOAL and GOAL+APPROACH permit the closest feasible analysis with explicit limitations.

Scoring protocol. Deterministic validators assess code executability, schema use, artifact completeness, temporal/grain consistency, and claim support. Study-specification validity is scored after mapping each system's submitted or parsed specification into the common benchmark schema; native ProCARE IR objects are not treated as ground truth. Residual semantic criteria are scored by Gemini-2.5-Pro and Qwen2.5-1M at temperature 0; judges are blinded to method identity and evaluate full bundles under fixed rubrics. The same validators, rubrics, and post-hoc diagnostics are applied to ProCARE, non-ProCARE baselines, and Public-RWS; Public-RWS omits repeated board-certified review. All scores are normalized to $[0, 1]$, and automated diagnostics are compared with expert review on the primary benchmark. Artifact-level metrics support cross-system comparisons. Contract-native metrics, such as explicit IR validity, native certificate coverage, and report-DAG consistency, are reported as mechanism diagnostics and are not interpreted alone as evidence of clinical, causal, or statistical correctness.

Let N_v, R_v be detected and resolved execution-time violations, and N_q, C_q, U_q, L_q be factual, certified, unsupported, and language-violating claims. We compute

$$\mathrm{VRR} = \frac{R_v}{N_v}, \qquad \mathrm{CTC} = \frac{C_q}{N_q}, \qquad \mathrm{UCR} = \frac{U_q}{N_q}, \qquad \mathrm{CLV} = \frac{L_q}{N_q}. \tag{5}$$

VRR is measured over execution-time violations before the accepted run; CTC, UCR, and CLV are measured over final-report factual claims. To reduce evaluator coupling, native contract artifacts

Table 2: Evaluation metrics for end-to-end RWS automation.

Dimension	Metrics and scoring
IR	**GA**: LLM-rubric goal alignment; **PF**: schema/IR feasibility with LLM adjudication.
Execution	**XE**: sandbox pass rate n_{pass}/N; **CQ**: LLM-rubric code quality; **VRR**: resolved/detected contract violations.
Visualization	**VC**: LLM-rubric visual clarity; **CF**: chart–result consistency.
Reporting	**RS**: report schema and section coverage; **SL**: LLM-rubric scientific language.
Claims	**CTC**: post-hoc claim trace coverage; **UCR**: unsupported factual-claim rate; **CLV**: language-support violation rate.

are separated from artifact-level scoring where possible. ProCARE's RWS-IRs, manifests, verifier traces, and evidence certificates are treated as system outputs rather than ground truth. For non-ProCARE systems, the checker reconstructs the closest available study specification, artifact manifest, and claim-evidence links from code, logs, tables, figures, reports, profile entries, and recorded assumptions. CTC/UCR/CLV are then scored post hoc against executed artifacts, rather than by accepting native certificates. Native certificate and DAG diagnostics are retained only for mechanism audit. This mitigates, but does not eliminate, coupling because validity is still operationalized through profile-derived constraints.

Primary HCC benchmark. The primary benchmark is derived from IRB-approved longitudinal hepatocellular carcinoma data collected at a tertiary medical center. The released version is a fully de-identified record-level benchmark: direct identifiers and fields that could identify individual participants are removed before publication, while random benchmark subject and visit identifiers preserve the relational structure needed for reproduction. The cohort contains 1,295 subjects, 12,847 visits, and 54 interlinked electronic case-report-form sheets covering baseline intake, follow-up, treatments, laboratory measurements, outcomes, and quality-of-life assessments. The dataset captures practical RWS challenges: subject- and visit-level linkage, sparse high-dimensional forms, form-specific clinical conventions, redundant administrative fields, and longitudinal structure for repeated-measures and time-to-event analyses. The public benchmark includes the de-identified HCC tables, task specifications, schema/profile artifacts, prompts, data dictionary, checksums, and evaluation scripts; a complete sheet inventory and release description are provided in the appendix.

Public-RWS external validation suite. Public-RWS-18 evaluates cross-dataset transfer on three public health datasets: UCI Diabetes 130-US Hospitals, an EHR-derived encounter-level readmission dataset [36, 37]; a NHANES 2017–2018 subset linking demographics, examination, questionnaire, laboratory, medical-condition, and prescription-medication tables by `SEQN` [38]; and TCGA-LIHC, HCC-aligned oncology metadata with demographic, diagnosis, treatment, vital-status, and follow-up-related fields [39]. Each dataset contributes two task objectives under three prompt-specificity levels, yielding 18 external validation tasks. The suite tests schema shift, multi-table grounding, proxy-outcome handling, and claim traceability, complementing the primary HCC benchmark.

5 Experiments

We evaluate ProCARE along five research questions: **RQ1** end-to-end artifact validity against LLM-agent and table-analysis baselines; **RQ2** contributions of individual contract mechanisms; **RQ3** claim-level traceability and agreement with a five-member clinical/statistical expert panel; **RQ4** robustness across RWS objectives, prompt specificity, and public-data schema shifts; and **RQ5** sensitivity to the base LLM backbone and profiling cost amortization. The evaluation spans 36 primary HCC RWS tasks, 18 Public-RWS external validation tasks, baseline comparisons, ablations, claim diagnostics, expert review, backbone sensitivity, and cost analysis.

5.1 Main Results

Baseline comparison (RQ1). We first compare ProCARE with LLM-agent, table-analysis, and code-agent baselines under matched prompts, inference budgets, and artifact contracts. Table 3 shows that existing methods often either fail to produce executable artifact bundles or emit factual claims unsupported by admissible artifacts, code, or claim policies. Claude Code and Codex substantially improve executability and code quality over generic agents, but remain weaker than ProCARE on

Table 3: Baseline comparison on 36 HCC RWS tasks. GA–SL report task performance, while VRR–CLV report contract diagnostics following Table 2; N/A indicates no executable artifact bundle. †Proprietary; ‡Open-source; §Agent-based.

Method	GA↑	PF↑	XE↑	CQ↑	VC↑	CF↑	RS↑	SL↑	VRR↑	CTC↑	UCR↓	CLV↓
ReAct‡§	0.78	0.62	0.38	0.47	0.28	0.48	0.63	0.59	0.34	0.48	0.24	0.17
ToolLLM‡§	0.73	0.68	0.00	N/A	N/A	N/A	0.56	0.24	0.00	0.21	0.33	0.24
TableGPT-Agent‡§	0.61	0.59	N/A	N/A	N/A	N/A	0.40	0.43	N/A	0.18	0.36	0.22
ChatExcel-Max†§	N/A	N/A	N/A	N/A	N/A	N/A	0.52	0.27	N/A	0.12	0.39	0.28
Gemini-3-Pro direct†	0.75	0.51	0.13	0.45	0.28	0.48	0.61	0.53	0.18	0.36	0.27	0.15
TableGPT2‡	0.41	0.45	0.00	0.25	N/A	N/A	N/A	N/A	0.00	N/A	N/A	N/A
Claude Code†§	0.82	0.75	0.86	0.80	0.52	0.58	0.71	0.66	0.70	0.66	0.16	0.12
Codex†§	0.83	0.77	0.89	0.82	0.55	0.61	0.73	0.68	0.73	0.69	0.14	0.10
Methods without valid artifact bundles: LLM (DeepSeek-R1-0528); ChatExcel-Pro†§.												
ProCARE (Ours)	**0.86**	**0.84**	**0.92**	**0.88**	**0.72**	**0.74**	**0.80**	**0.79**	**0.81**	**0.90**	**0.06**	**0.03**

Table 4: Scaffolded RQ1 controls. Direct rows use Gemini-3-Pro with the same decoding settings as the default ProCARE backbone. Non-ProCARE rows are independent baseline adaptations; ProCARE modules are not transplanted into them.

Baseline	Fairness scaffold	PF↑	XE↑	CQ↑	VRR↑	CTC↑	UCR↓	CLV↓
ReAct + scaffold	Φ summary + bundle tmpl. + val. feedback + 3 repairs	0.79	0.74	0.69	0.62	0.70	0.13	0.10
ToolLLM + scaffold	Same scaffold on the tool-use baseline	0.80	0.68	0.65	0.58	0.66	0.15	0.12
G3P + profile/template	Φ + manifests; no validator loop	0.73	0.66	0.62	0.31	0.55	0.20	0.15
G3P + validator repair	Φ + manifests + val. feedback + repairs	0.78	0.79	0.72	0.68	0.65	0.15	0.11
G3P + study-spec schema	Emits study-spec artifact; no IR compiler	0.82	0.70	0.68	0.43	0.52	0.21	0.16
G3P + claim-trace schema	Emits claim-trace artifact; no cert. gate	0.74	0.63	0.61	0.35	0.74	0.11	0.08
ProCARE reference	**Profile–IR–validator–certificate–DAG workflow**	**0.84**	**0.92**	**0.88**	**0.81**	**0.90**	**0.06**	**0.03**

Table 5: Mechanism ablation over 36 RWS instances. N/A indicates no typed RWS-IR.

Variant	PF↑	XE↑	VRR↑	CTC↑	UCR↓	SL↑	Dominant failure
No profile Φ	0.49	0.18	0.35	0.58	0.28	0.42	Field hallucination; invalid joins.
No RWS-IR	N/A	0.34	0.44	0.63	0.22	0.69	Missing grain, anchors, and claim policy.
No repair	0.84	0.69	0.43	0.80	0.13	0.70	Unresolved runtime and artifact violations.
No certificates	0.84	0.92	0.81	N/A	0.26	0.62	Unsupported causal or comparative claims.
No DAG ordering	0.84	0.92	0.81	0.84	0.11	0.64	Dependency-agnostic report ordering.
Full ProCARE	**0.84**	**0.92**	**0.81**	**0.90**	**0.06**	**0.79**	Localized repair; traceable final claims.

clinically grounded profile fit, variable-role consistency, statistical language, and claim traceability. To partially control for evaluator bias toward systems that natively emit IRs, manifests, and evidence certificates, we additionally evaluate scaffolded baselines. These RQ1 controls keep the tested system as an external baseline or direct Gemini-3-Pro run, while only adding evaluator-facing information, output schemas, validator feedback, and repair budget. They do not expose ProCARE's typed IR compiler, certificate admission gate, certified reporting DAG, or internal planner state. The scaffolds improve generic baselines, especially in executability and artifact completeness. ProCARE still achieves the strongest violation resolution and post-hoc claim traceability (Table 4), suggesting that its advantage is not explained by schema exposure or manifest formatting alone.

Contract mechanism ablation (RQ2). To identify the sources of improvement, we ablate each contract object in Table 5. The profile yields the largest contribution, suggesting that reliable schema, field, and join constraints are the basis for valid execution and downstream reasoning. RWS-IR supplies the typed study contract and strengthens claim-policy control. Violation-guided repair mainly improves executable constraint satisfaction, whereas evidence certificates and certified DAG ordering primarily affect reporting: the former enforces claim traceability, and the latter preserves dependency-aware structure. These results indicate that the framework's gains come from complementary contract objects: structural contracts support computation, repair improves execution robustness, and evidence/order contracts regulate faithful reporting.

Claim traceability and expert validation (RQ3). Table 6 reports post-hoc claim diagnostics for ProCARE. More specific prompts improve CTC and reduce UCR/CLV, mainly by constraining

Table 6: ProCARE claim diagnostics. Values are averaged over 12 instances per prompt level; Overall aggregates all 36 instances.

Prompt	Claims	CTC↑	UCR↓	CLV↓	Main downgrade reason
GOAL	5.8	0.87	0.09	0.05	Unsupported causal/comparative wording.
GOAL+APPROACH	6.4	0.90	0.05	0.03	Proxy endpoint beyond evidence level.
GOAL+METHOD	6.9	0.94	0.03	0.02	Sparse endpoint or subgroup evidence.
Overall	**6.4**	**0.90**	**0.06**	**0.03**	Downgrades mainly concern causal strength.

Table 7: Expert consistency of automated metrics. Expert scores are mean±SD on a 5-point scale; CI and reliability details are reported in the appendix.

Stage	Reviewers	Metric(s)	Auto	Expert	ρ
RWS-IR	3 clinical + 2 stats/data	$(GA + PF)/2$	0.82	4.38±0.48	0.82
Code	2 stats/data	CQ	0.88	4.42±0.21	0.74
Visualization	3 clinical + 2 stats/data	$(VC + CF)/2$	0.66	4.50±0.53	0.75
Reporting	3 clinical + 2 stats/data	SL	0.76	4.13±0.64	0.71

All Spearman correlations are computed across the 36 primary HCC instances; all $p < 0.001$.

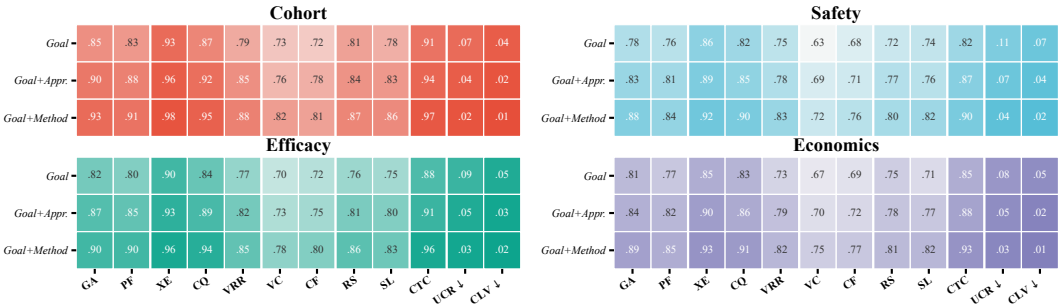


Figure 2: Performance across four RWS tasks and prompt-specificity levels. Cell labels report raw metric values; UCR and CLV are lower-is-better rates with reversed color intensity.

variable bindings, temporal anchors, and methods. Remaining downgrades mostly involve causal or high-stakes wording unsupported by proxy artifacts, sparse endpoints, or subgroup evidence. We further compare automated diagnostics with blinded review by three clinical and two statistics/data experts. Table 7 shows moderate-to-strong Spearman agreement across stages, strongest for RWS-IR validity ($\rho = 0.82$) and significant for code, visualization, and reporting. The appendix provides reviewer rubrics, aggregation rules, confidence intervals, and inter-rater reliability.

5.2 Robustness and Deployment Analyses

Task, prompt, and public-data robustness (RQ4). Figure 2 evaluates ProCARE across four RWS objectives and three prompt-specificity levels on the HCC benchmark. Metrics generally improve with prompt specificity, suggesting that methodological context helps specification, repair, and claim control. Performance is stable across task families; residual variation is mainly due to sparse endpoints, limited subgroup support, and claim-scope restrictions. Table 8 evaluates Public-RWS-18 schema transfer without dataset-specific changes. Claim policies restrict outputs to descriptive/readmission-proxy claims for UCI Diabetes, descriptive/association claims for NHANES, and descriptive oncology-metadata/follow-up-proxy claims for TCGA-LIHC. ProCARE outperforms Claude Code and Scaffolded G3P in violation resolution, post-hoc claim traceability, UCR, and CLV; NHANES remains hardest due to multi-table survey joins.

Backbone sensitivity and cost (RQ5). Holding the profile, RWS-IR, validators, and certificate gate fixed, ProCARE remained contract-consistent across tested backbones; stronger planners mainly improved GA, PF, VRR, and CTC, while execution metrics were comparatively stable. Table 9

Table 8: Public-RWS-18 transfer results. Values average six tasks per dataset; baseline claim traces are computed post hoc.

Dataset	Method	XE↑	VRR↑	CTC↑	UCR↓	CLV↓
UCI Diabetes	Claude Code	0.60	0.61	0.69	0.12	0.09
	Scaffolded G3P	0.57	0.59	0.66	0.13	0.10
	ProCARE	**0.64**	**0.72**	**0.87**	**0.07**	**0.04**
NHANES	Claude Code	0.52	0.54	0.63	0.15	0.11
	Scaffolded G3P	0.50	0.52	0.60	0.16	0.12
	ProCARE	**0.58**	**0.66**	**0.84**	**0.09**	**0.06**
TCGA-LIHC	Claude Code	0.57	0.58	0.66	0.14	0.10
	Scaffolded G3P	0.54	0.56	0.63	0.15	0.11
	ProCARE	**0.61**	**0.69**	**0.86**	**0.08**	**0.05**

Table 9: Backbone sensitivity with fixed ProCARE workflow. Metrics follow Table 2; model access details and version identifiers are reported in the appendix. †Proprietary models; ‡Open-source models.

Family	Model	Performance								Contract diagnostics			
		GA↑	PF↑	XE↑	CQ↑	VC↑	CF↑	RS↑	SL↑	VRR↑	CTC↑	UCR↓	CLV↓
GPT	GPT-5.2†	0.84	0.82	0.94	0.84	0.68	0.70	0.78	0.77	0.78	0.88	**0.06**	0.04
	GPT-4 Turbo†	0.81	0.78	0.89	0.80	0.64	0.66	0.75	0.74	0.74	0.85	0.08	0.06
Claude	Claude Opus 4.5†	0.85	0.83	**0.97**	0.86	**0.72**	**0.74**	0.78	0.77	0.80	0.89	**0.06**	0.04
	Claude Sonnet 4.5†	0.83	0.81	0.94	0.84	0.70	0.72	0.76	0.75	0.78	0.87	0.07	0.05
Gemini	Gemini-3-Pro†	**0.86**	**0.84**	0.92	**0.88**	**0.72**	**0.74**	**0.80**	**0.79**	**0.81**	**0.90**	**0.06**	**0.03**
	Gemini-2.5-Pro†	0.82	0.78	0.91	0.82	0.66	0.68	0.75	0.74	0.76	0.86	0.08	0.06
Qwen	Qwen3-Max†	0.77	0.70	0.84	0.76	0.58	0.60	0.68	0.68	0.68	0.80	0.12	0.09
	Qwen3-235B-Instruct‡	0.74	0.66	0.80	0.72	0.55	0.57	0.64	0.65	0.64	0.76	0.15	0.11
DeepSeek	DeepSeek-V3.2‡	0.76	0.68	0.86	0.78	0.60	0.62	0.66	0.67	0.70	0.80	0.11	0.08
	DeepSeek-R1-0528‡	0.72	0.64	0.82	0.74	0.56	0.58	0.62	0.63	0.66	0.77	0.13	0.10

reports the backbone comparison; model access details and version identifiers are provided in the appendix.

We report token usage for ProCARE; differences in agent pipelines limit direct cost comparisons with the baselines. Profiling the HCC dataset requires approximately $T_{\text{prof}} = 65$k tokens once, and each subsequent query uses approximately $T_{\text{inf}} = 38$k tokens. Across N queries on the same dataset, the average token cost is $\bar{T}(N) = T_{\text{inf}} + T_{\text{prof}}/N$, decreasing from approximately 103k tokens for one query to 41.3k tokens per query for 20 queries.

Discussion and Limitations. Overall, ProCARE should be interpreted as a contract layer for making RWS assumptions, artifacts, and claim support auditable under human oversight. Its reliability depends on the correctness of the constructed profile, the coverage of deterministic validators, and the assumptions encoded in the RWS-IR. A successful run shows that the generated bundle satisfies the implemented schema, method, temporal, and artifact checks; it does not establish clinical optimality, adequate confounding control, causal validity, or evidentiary sufficiency beyond the recorded assumptions. The primary benchmark is single-center and disease-specific, and Public-RWS-18 mainly tests schema transfer rather than multi-institutional longitudinal generalization. Deployments on OMOP, FHIR, or institutional schemas therefore require re-instantiating the profile and revalidating the IR, execution gates, and certificate checks.

6 Conclusion

We introduced ProCARE, a profile-grounded contract architecture for RWS automation. ProCARE compiles study goals into typed RWS-IR, generates executable artifact bundles through a profile-constrained coding harness, repairs localized contract violations, and admits report claims through evidence certificates. On a 36-instance HCC RWS benchmark and Public-RWS-18 external validation

suite, these mechanisms improve executable validity, code quality, and claim traceability relative to LLM-agent and table-analysis baselines. The public release includes the fully de-identified HCC benchmark tables and scoring assets to support independent reproduction. Future work should test the same contract mechanisms on multi-institutional longitudinal RWS data and standardized OMOP/FHIR deployments.

Acknowledgments and Disclosure of Funding

This work was supported by the Science and Technology Commission of Shanghai Municipality (Grant No. 24DP1500704) and the "Medical+X" Interdisciplinary Research Project of Tongji University (Grant No. 2025-0674-YB-02).

References

[1] Rachel E. Sherman, Steven A. Anderson, Gerald J. Dal Pan, Gerry W. Gray, Thomas Gross, Nina L. Hunter, Lisa LaVange, Danica Marinac-Dabic, Peter W. Marks, Melissa A. Robb, Jeffrey Shuren, Robert Temple, Janet Woodcock, Lilly Q. Yue, and Robert M. Califf. Real-world evidence - what is it and what can it tell us? *The New England journal of medicine*, 375 (23):2293–2297, 2016. doi: 10.1056/NEJMsb1609216.

[2] Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Anh T. N. Nguyen, Lauren T. May, Geoffrey I. Webb, and Shirui Pan. Large language models for scientific synthesis, inference and explanation. *arXiv preprint arXiv:2310.07984*, 2023.

[3] Yixing Jiang, Kameron C Black, Gloria Geng, Danny Park, James Zou, Andrew Y Ng, and Jonathan H Chen. MedAgentBench: A realistic virtual EHR environment to benchmark medical LLM agents. *arXiv preprint arXiv:2501.14654*, 2025.

[4] Jonathan Kim, Anna Podlasek, Kie Shidara, Feng Liu, Ahmed Alaa, and Danilo Bernardo. Limitations of large language models in clinical problem-solving arising from inflexible reasoning. *Scientific Reports*, 15(1):39426, 2025.

[5] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew Dai, Nissan Hajaj, Peter Liu, Xiaobing Liu, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Gavin Duggan, Gerardo Flores, Michaela Hardt, Jamie Irvine, Quoc Le, Kurt Litsch, Jake Marcus, Alexander Mossin, and Jeff Dean. Scalable and accurate deep learning for electronic health records. *npj Digital Medicine*, 1:18, 2018. doi: 10.1038/s41746-018-0029-1.

[6] George Hripcsak, Jon Duke, Nigam Shah, Christian Reich, Vojtech Huser, Martijn Schuemie, Marc Suchard, Rae Woong Park, Ian Wong, Peter Rijnbeek, Johan Lei, Nicole Pratt, Niklas Norén, Yu-Chuan Li, Paul Stang, David Madigan, and Patrick Ryan. Observational health data sciences and informatics (ohdsi): Opportunities for observational researchers. *Studies in health technology and informatics*, 216:574–8, 2015.

[7] Jan-Micha Bodensohn, Ulf Brackmann, Liane Vogel, Anupam Sanghi, and Carsten Binnig. Unveiling challenges for llms in enterprise data engineering. *arXiv preprint arXiv:2504.10950*, 2025.

[8] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *Nature*, 620:172–180, 2023.

[9] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne J. Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun K. Manrai, Debora S. Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Velickovic, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka

Zitnik. Scientific discovery in the age of artificial intelligence. *Nature*, 620:47–60, 2023. URL https://api.semanticscholar.org/CorpusID:260384616.

[10] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

[11] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[12] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[13] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the 12th International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

[14] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

[15] Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. InfiAgent-DABench: Evaluating agents on data analysis tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 19544–19572. PMLR, 2024. URL https://proceedings.mlr.press/v235/hu24s.html.

[16] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Yongxin Ni, Zhibin Gou, Zongze Xu, Yuyu Luo, and Chenglin Wu. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1016. URL https://aclanthology.org/2025.findings-acl.1016/.

[17] Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. DSBench: How far are data science agents from becoming data science experts? In *Proceedings of the 13th International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DSsSPr0RZJ.

[18] Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. DataSciBench: An LLM agent benchmark for data science, 2025. URL https://arxiv.org/abs/2502.13897. Use cautiously in main text; the corresponding ICLR 2025 OpenReview submission was withdrawn.

[19] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. SheetCopilot: Bringing software productivity to the next level through large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=tfyr2zRVoK.

[20] Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. Encoding spreadsheets for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language*

Processing, pages 20728–20748, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1154. URL https://aclanthology.org/2024.emnlp-main.1154/.

[21] Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long, Liyao Li, Pengzuo Wu, Qi Zhang, Qingyi Huang, Saisai Yang, Tao Zhang, Wentao Ye, Wufang Zhu, Xiaomeng Hu, Xijun Gu, Xinjie Sun, Xiang Li, Yuhang Yang, and Zhiqing Xiao. TableGPT2: A large multimodal model with tabular data integration, 2024. URL https://arxiv.org/abs/2411.02059.

[22] Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. TableRAG: Million-token table understanding with language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=41lovPOCo5.

[23] Xiaohan Yu, Pu Jian, and Chong Chen. TableRAG: A retrieval augmented generation framework for heterogeneous document reasoning, 2025. URL https://arxiv.org/abs/2506.10380. Accepted by EMNLP 2025.

[24] Gyubok Lee, Sunjun Kweon, Seongsu Bae, and Edward Choi. Overview of the EHRSQL 2024 shared task on reliable text-to-SQL modeling on electronic health records. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 644–654, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.clinicalnlp-1.62. URL https://aclanthology.org/2024.clinicalnlp-1.62/.

[25] Xiao Yang, Xuejiao Zhao, and Zhiqi Shen. EHRStruct: A comprehensive benchmark framework for evaluating large language models on structured electronic health record tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 34340–34348, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/40731.

[26] Gyubok Lee, Elea Bach, Eric Yang, Tom Pollard, Alistair Johnson, Edward Choi, Yugang Jia, and Jong Ha Lee. FHIR-AgentBench: Benchmarking LLM agents for realistic interoperable EHR question answering, 2025. URL https://arxiv.org/abs/2509.19319.

[27] Jan P. Vandenbroucke, Erik von Elm, Douglas G. Altman, Peter C. Gøtzsche, Cynthia D. Mulrow, Stuart J. Pocock, Charles Poole, James J. Schlesselman, and Matthias Egger. Strengthening the reporting of observational studies in epidemiology (STROBE): Explanation and elaboration. *PLoS Medicine*, 4(10):e297, 2007. doi: 10.1371/journal.pmed.0040297. URL https://doi.org/10.1371/journal.pmed.0040297.

[28] Eric I. Benchimol, Liam Smeeth, Astrid Guttmann, Katie Harron, David Moher, Irene Petersen, Henrik T. Sørensen, Erik von Elm, and Sinéad M. Langan. The REporting of studies conducted using observational routinely-collected health data (RECORD) statement. *PLoS Medicine*, 12(10):e1001885, 2015. doi: 10.1371/journal.pmed.1001885. URL https://doi.org/10.1371/journal.pmed.1001885.

[29] Miguel A. Hernán and James M. Robins. Using big data to emulate a target trial when a randomized trial is not available. *American Journal of Epidemiology*, 183(8):758–764, 2016. doi: 10.1093/aje/kwv254. URL https://doi.org/10.1093/aje/kwv254.

[30] Aidan G. Cashin, Harrison J. Hansford, Miguel A. Hernán, Sonja A. Swanson, Hopin Lee, Matthew D. Jones, Issa J. Dahabreh, Barbra A. Dickerman, Matthias Egger, Xabier Garcia-Albeniz, Robert M. Golub, Nazrul Islam, Sara Lodi, Margarita Moreno-Betancur, Sallie-Anne Pearson, Sebastian Schneeweiss, Melissa K. Sharp, Jonathan A. C. Sterne, Elizabeth A. Stuart, and James H. McAuley. Transparent reporting of observational studies emulating a target trial—the TARGET statement. *JAMA*, 334(12):1084–1093, 2025. doi: 10.1001/jama.2025.13350. URL https://doi.org/10.1001/jama.2025.13350.

[31] Observational Health Data Sciences and Informatics. OMOP common data model. https://ohdsi.github.io/CommonDataModel/, 2026. Accessed: 2026-05-02.

[32] Health Level Seven International. FHIR release 4.0.1: Fast healthcare interoperability resources. https://www.hl7.org/fhir/, 2019. Accessed: 2026-05-02.

[33] U.S. Food and Drug Administration. Data standards for drug and biological product submissions containing real-world data: Guidance for industry. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/data-standards-drug-and-biological-product-submissions-containing-real-world-data, 2023. Final guidance, December 2023. Accessed: 2026-05-02.

[34] Dubai Li, Yuxiang He, Yan Hu, Yu Tian, and Jingsong Li. Can LLM agents generate real-world evidence? evaluating observational studies in medical databases, 2026. URL https://arxiv.org/abs/2603.22767.

[35] Nayoon Gim, In Gim, Yu Jiang, Yuka Kihara, Marian Blazes, Yue Wu, Cecilia S. Lee, and Aaron Y. Lee. An LLM-assisted framework for accelerated and verifiable clinical hypothesis testing from electronic health records. *medRxiv*, 2026. doi: 10.64898/2026.02.10.26346008. URL https://www.medrxiv.org/content/10.64898/2026.02.10.26346008v1. Preprint.

[36] John Clore, Krzysztof J. Cios, Jonathan P. DeShazo, and Beata Strack. Diabetes 130-US hospitals for years 1999–2008, 2014. URL https://archive.ics.uci.edu/dataset/296/diabetes%2B130-us%2Bhospitals%2Bfor%2Byears%2B1999-2008.

[37] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Research International*, 2014:781670, 2014. doi: 10.1155/2014/781670. URL https://doi.org/10.1155/2014/781670.

[38] Centers for Disease Control and Prevention, National Center for Health Statistics. NHANES 2017–2018 questionnaires, datasets, and related documentation. https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2017, 2020. Accessed: 2026-05-03.

[39] National Cancer Institute Genomic Data Commons. TCGA-LIHC: Liver hepatocellular carcinoma clinical metadata. https://portal.gdc.cancer.gov/projects/TCGA-LIHC, 2026. Accessed: 2026-05-03.

A Appendix Roadmap, Navigation, and Main-Paper Coverage

The appendix contains both claim-support material for the main paper and standalone audit material that is not fully described in the main text. Table 10 gives a complete section-level map of the supplement. Table 11 then maps main-paper claims to their most relevant supporting material.

Table 10: Complete appendix navigation. The map covers all supplement sections, including audit and interface material that is not tied to a specific main-paper table or figure.

Appendix section	Contents	Purpose in the supplement
App. A	Roadmap, complete navigation, and main-paper crosswalk.	Orients readers to both the full supplement and the subset of material that directly supports main-paper claims.
App. B	HCC benchmark scope, release boundary, task construction, Public-RWS-18, claim policies, and worksheet inventory.	Documents the datasets, task instances, and schema constraints needed to reproduce or audit the benchmark.
App. C	Profile schema, concrete profile instance, RWS-IR checks, evidence certificates, validators, artifact bundle, coding harness, and DAG reporting.	Specifies the executable contract objects and implementation checks behind ProCARE.
App. D	Execution environment, cost accounting, baseline adaptation, scaffolded fairness controls, ablations, model access, and repository commands.	Records experimental controls and reproducibility details.
App. E	Claim-diagnostic interpretation and backbone-sensitivity interpretation.	Explains claim-policy downgrades and performance differences across backbones.
App. F	Dashboard, profile panel, RWS-IR view, coding loop, report review, and section-wise report generation screenshots.	Supplies workflow/interface evidence showing where artifacts, traces, logs, and evidence-linked report sections are inspected.
App. G	Metric definitions, deterministic validator rules, LLM judge protocol, expert-review protocol, raw expert-data schema, reliability, and rubric anchors.	Defines how automated and human evaluation scores are produced and audited.
App. H	Benchmark, IR compiler, coding harness, report/certificate, and judge prompt templates.	Makes the generation and scoring prompts inspectable.
App. I	Data release, privacy, responsible-use caveats, and deployment limits.	Clarifies the de-identified HCC release, public-data sources, and how the system should be interpreted.

Table 11: Main-paper claims and corresponding supplementary support. This crosswalk is not the full appendix map; complete section navigation is provided in Table 10.

Main-paper location	Claim or reference	Supplementary support
Sec. 3.1	The heterogeneous profile records structural and knowledge-driven constraints, with a concrete serialization and de-identified profile instance.	App. C.1 and App. C.2.
Sec. 3.2	RWS-IR compilation and execution enforce schema, semantic-role, temporal, grain, method, and artifact constraints.	App. C.3, App. C.5, and App. C.10.
Sec. 3.3	Evidence certificates and claim policies gate factual report claims, while a DAG orders supported report sections.	App. C.6, App. C.9, and App. C.11.
Sec. 4	The primary benchmark contains 36 HCC tasks over four objectives and a 54-sheet longitudinal eCRF workbook.	App. B and Table 23.
Sec. 4–5	Public-RWS-18 uses three public datasets, dataset-specific claim policies, and the same validators and rubrics.	App. B.6.
Main Table 3	LLM-agent, table-analysis, and code-agent baselines are evaluated under a unified prompt, budget, artifact contract, and post-hoc checker.	App. D.2.
Main Table 5	Ablations remove one contract mechanism at a time while keeping the rest of the workflow fixed when defined.	App. D.4.
Main Table 6	Claim diagnostics are computed over factual report claims with trace and policy checks.	App. E.1.
Main Table 7	Automated metrics are compared with expert review.	App. G.4.
Main Figure 2	Stage-wise performance varies by objective and prompt specificity.	App. B and App. G.
Main Table 9 and Sec. 5.2	Backbone sensitivity uses fixed workflow components; model metadata and amortized token cost are reported.	App. D.5 and App. D.1.

Table 12: HCC benchmark release contents and privacy treatment.

Release item	Included material	Privacy and reproducibility treatment
De-identified HCC tables	54 linked eCRF sheets covering the 1,295-subject benchmark cohort.	Direct participant identifiers are removed; random benchmark IDs preserve subject-level and visit-level linkage required for rerunning all tasks.
Task and prompt files	The 36 primary HCC benchmark prompts and task metadata.	Prompts refer only to de-identified table and field names and do not expose participant identities.
Schema/profile artifacts	Field dictionaries, join keys, temporal roles, missingness summaries, and profile objects.	Contains schema and aggregate metadata used by validators; no direct identity fields are present.
Evaluation assets	Validators, judge prompts, artifact templates, scoring scripts, and expected output schema.	Enables independent reruns against the released de-identified tables.
Release metadata	Data dictionary, checksums, and responsible-use terms.	Documents provenance, file integrity, de-identification scope, and restrictions against re-identification or clinical use.

B Benchmark Datasets and Tasks

This section gives the benchmark details not repeated in the main paper: release contents, worksheet organization, task instantiation, public-data transfer tasks, and the worksheet inventory used by the validators.

B.1 Supplemental schema scope

The main paper summarizes the cohort size and high-level eCRF scope. Here we record the schema properties needed to reproduce profiling and task construction. The workbook is multi-table by design, spanning enrollment, baseline characterization, time-varying interventions, follow-up, and survival outcomes; this structure is what drives the join, grain, temporal-anchor, and claim-policy checks used throughout the benchmark.

B.2 Release boundary and de-identification

The public benchmark release contains the fully de-identified HCC record-level tables used in the 36 primary tasks, together with task specifications, prompt files, schema and profile artifacts, evaluation scripts, aggregate profile examples, checksums, and a data dictionary. Before release, direct identifiers and fields that could identify individual participants are removed from the benchmark copy. Random benchmark subject and visit identifiers are retained only to preserve within-dataset linkage, visit order, and longitudinal analysis structure. The released tables are intended for reproducibility of the benchmark and do not constitute a clinical deployment dataset.

B.3 Functional groupings

Table 13 groups worksheets by their role in profiling and benchmark construction.

B.4 Dataset limitations for benchmark design

Several data properties directly shape the benchmark tasks. Adverse-event screening pages are available, but detailed AE/SAE records are sparse in the released workbook; safety tasks therefore distinguish screening-level summaries from detailed event-term analyses. The dataset does not contain direct billing or price fields, so economics-oriented tasks are framed as resource utilization, follow-up burden, or quality-of-life proxy analyses rather than cost-effectiveness studies.

Table 13: Functional worksheet groups used by profiling and benchmark construction.

Group	Representative sheets	Main contents	Use in benchmark/profile
System metadata	SYSVAR, ECRF, DATA_RANGE	Form catalog, site allocation, variable descriptions, and codebook fields.	Schema parsing, worksheet indexing, and field-name grounding.
Baseline	DM, PH, CP, BCLC, ECOG	Demographics, history, liver function, staging, and performance status.	Cohort definition, baseline table construction, and confounder checks.
Longitudinal labs	LB_AFP, LB_BB, LB_BR, LB_TT	Tumor markers, biochemistry, complete blood count, and coagulation tests.	Safety proxy signals and follow-up burden measures.
Treatment	TACE, TART, IMU, PATE, CM	Procedures, systemic therapy, retreatment, and concomitant medication records.	Exposure definition, treatment intensity, and resource-utilization tasks.
Outcomes	VT, AE_HQ, SAE_HQ, SF	Recurrence and survival proxies, AE/SAE screening, and quality-of-life forms.	Endpoint derivation and limitation checks.

B.5 Benchmark task construction

The benchmark crosses four RWS objectives, three core questions per objective, and three prompt-specificity levels:

$$|\mathcal{B}| = |\mathcal{O}| \times |\mathcal{Q}| \times |\mathcal{L}| = 4 \times 3 \times 3 = 36.$$

Table 14 defines the prompt levels used across all objectives.

For each metric in Main Figure 2, an objective–prompt cell averages the three benchmark instances for that objective at the specified prompt level. A fixed prompt level therefore contains twelve instances across the four objectives; the full benchmark contains all 36 instances across the three levels.

Table 14: Prompt specificity levels in the benchmark.

Level	Information provided	Expected behavior
GOAL	High-level study intent only.	Select feasible variables and disclose assumptions.
GOAL+APPROACH	Study intent plus an analysis strategy.	Follow the strategy when the data support it; otherwise report the deviation.
GOAL+METHOD	Specific variables, time windows, or statistical method.	Stop or report infeasibility when required data are missing.

All instances use the same output contract: a verification plan, executable analysis entry point, generated artifacts, and a structured RWS report. Table 15 lists the twelve core questions, and Tables 16–19 give the corresponding 36 prompts.

B.6 Public-RWS external validation suite

Public-RWS-18 is an external validation suite for testing whether the same profile–IR–certificate pipeline transfers to public health datasets with different schemas and study designs. It is not merged with the 36-instance HCC benchmark and is not treated as another full longitudinal RWS benchmark. Each public dataset contributes two task objectives, and each objective is evaluated under low-, medium-, and high-specificity prompts.

Public-RWS-18 uses the same profile construction, RWS-IR schema, validators, and claim checker as the HCC benchmark. Dataset-specific preprocessing is limited to loading the public tables, normalizing file and column names for stable tool access, preserving original public codebook variables, and constructing profile metadata. No public dataset receives a custom validator, claim checker, or reporting rule. The claim policy in Table 22 is the only dataset-specific adaptation, because each public source supports a different study-design boundary.

Each public dataset contributes two objectives, and each objective is instantiated under three prompt-specificity levels. Dataset-level results are reported once in the main paper. Public-RWS-18 uses ReAct as a generic agent baseline and adds two stronger controls aligned with the primary baseline table: Claude Code as a native code-agent baseline, and Scaffolded G3P as the direct Gemini-3-Pro profile/template plus validator-repair control. Scaffolded G3P receives evaluator-facing schema

Table 15: Core research questions used to instantiate benchmark prompts.

Objective	ID	Core research question	Primary data sources
Cohort	C1	Characterize the analyzable HCC cohort, including enrollment, baseline characteristics, and clinical staging.	CO, DM, IE, SG, BCLC, CP, ECOG
Cohort	C2	Assess longitudinal follow-up completeness and data density.	CPH_FT, VT, LB_AFP, LB_BB, LB_BR, ECOG
Cohort	C3	Assess data readiness for downstream endpoint analyses.	DM, SG, BCLC, CP, ECOG, VT, treatment sheets
Safety	S1	Summarize AE/SAE screening rates and the availability of detailed event records.	AE_HQ, SAE_HQ, AE, SAE
Safety	S2	Describe postoperative laboratory safety signals from liver function, hematology, and coagulation panels.	LB_BB, LB_BR, LB_TT, SG
Safety	S3	Compare safety proxy signals after major treatment exposures.	TACE, TART, IMU, PATE, AE_HQ, SAE_HQ, lab sheets
Efficacy	E1	Analyze recurrence and recurrence-free survival proxies after surgery.	SG, VT, BCLC, CP, ECOG, treatment sheets
Efficacy	E2	Analyze overall survival proxies and their relationship to treatment exposure.	SG, VT, DM, BCLC, CP, ECOG, treatment sheets
Efficacy	E3	Analyze response assessment and longitudinal tumor-marker changes.	VT, LB_AFP, AFP2, BCLC, treatment sheets
Health economics/resource use	H1	Describe postoperative treatment resource utilization and treatment intensity.	TACE, TART, IMU, PATE, CM, SG
Health economics/resource use	H2	Estimate follow-up and testing burden as a resource-use proxy.	CPH_FT, VT, LB_AFP, LB_BB, LB_BR, LB_TT, ECOG, SF
Health economics/resource use	H3	Use SF-12 availability and treatment intensity to assess QoL/resource-use feasibility.	SF, TACE, TART, IMU, PATE, VT

Table 16: Cohort benchmark instances. Suffixes denote prompt specificity: G = GOAL, A = GOAL+APPROACH, and M = GOAL+METHOD.

ID	Prompt text	Required or expected evidence
C1-G	Characterize the baseline hepatocellular carcinoma cohort available in the workbook. Deliver a cohort summary suitable for a real-world study report.	Discover DM, SG, BCLC, CP, ECOG, and subject identifiers.
C1-A	Characterize the baseline HCC cohort. Build a STROBE-style cohort table covering demographics, surgery, liver function, tumor stage, and performance status.	Use patient-level baseline records, report missingness, and avoid duplicate subject counting.
C1-M	Produce a baseline cohort table for subjects with valid Subject_Id. Join DM, SG, BCLC, CP, and baseline ECOG; summarize AGE, SEX, HEIGHT, WEIGHT, SGDAT, SGTYP, BCLC, CPRES, CPGRA, and ECOG.	Use one baseline row per subject; report variable-level missingness; do not impute incomplete AGE.
C2-G	Assess whether the dataset has sufficient longitudinal follow-up for real-world evidence generation.	Identify visit-level forms and repeated measurements.
C2-A	Assess longitudinal follow-up completeness across follow-up forms, laboratory panels, tumor-marker tests, ECOG assessments, and efficacy follow-up records.	Use CPH_FT, VT, laboratory sheets, and ECOG; produce visit-level summaries.
C2-M	Quantify longitudinal data completeness using CPH_FT, VT, LB_AFP, LB_BB, LB_BR, and ECOG; count distinct visits by Subject_Id, tabulate availability by Visit_No, and estimate follow-up duration from SG.SGDAT to the latest VT.VTDAT1 or VT.VTDAT2.	Use Subject_Id and Visit_No; report subjects with no valid follow-up date.
C3-G	Evaluate whether the cohort is fit for downstream RWS endpoint analyses.	Reason about exposure, outcome, confounder, and time-anchor variables.
C3-A	Evaluate endpoint-analysis readiness through a feasibility matrix covering identifiers, surgery date, treatment exposure, recurrence/death endpoints, and baseline confounders.	Include data limitations, not only descriptive counts.
C3-M	Build a data-readiness matrix by checking Subject_Id, SG.SGDAT, recurrence dates VT.PADHDAT/VT.PADRDAT, death dates VT.DEDAT1/VT.DEDAT2, treatments from TACE, TART, IMU, PATE, and confounders DM.SEX, DM.AGE, BCLC.BCLC, CP.CPGRA, ECOG.ECOG.	Flag sparse or unavailable fields, especially VT.DEDAT2; do not invent endpoint fields.

Table 17: Safety benchmark instances.

ID	Prompt text	Required or expected evidence
S1-G	Analyze the adverse-event and serious-adverse-event safety profile in the cohort.	Discover that AE/SAE screening pages exist but detailed event tables are empty in the current workbook.
S1-A	Analyze AE and SAE safety reporting. Start with screening pages, then check whether detailed AE/SAE records are available for term, onset date, severity, causality, outcome, and seriousness.	Document infeasibility of detailed AE/SAE term analysis when detail tables have no valid records.
S1-M	Summarize AE/SAE reporting completeness using `AE_HQ.AEYN` and `SAE_HQ.SAEYN`; inspect `AE.AETERM`, `AE.AESTDAT`, `AE.AESEV`, `AE.AEREL`, `AE.AESAE`, `SAE.SAESTDAT`, and `SAE.SAESEV`.	Compute screening prevalence by subject; terminate detail-level analysis if `AE` and `SAE` contain zero valid records.
S2-G	Identify laboratory safety signals after surgery.	Use longitudinal laboratory sheets rather than relying only on AE tables.
S2-A	Identify post-surgical laboratory safety signals by summarizing liver function, hematology, and coagulation markers over follow-up visits.	Use lab value/result-judgment pairs and visit dates; report abnormal-result flags when available.
S2-M	Generate a post-surgical laboratory safety profile anchored at `SG.SGDAT`; use `LB_BB.LBBBDAT`, `ALTRES`, `ASTRES`, `TBILRES`, `ALBRES`; `LB_BR.BRDAT`, `WBCRES`, `HBRES`, `PLTRES`; and `LB_TT.LBTTDAT`, `PTRES`, `INRRES`, `DDIMRES`.	Summarize baseline and postoperative windows; use available-case denominators and `_SIG` flags when numeric thresholds are unavailable.
S3-G	Compare safety patterns across major postoperative treatment exposures.	Identify TACE, targeted therapy, immunotherapy, and retreatment exposure sheets.
S3-A	Compare safety proxy signals by treatment exposure using TACE, targeted therapy, immunotherapy, and postoperative retreatment groups.	Compare AE/SAE screening and laboratory abnormalities descriptively; avoid causal claims.
S3-M	Compare treatment-associated safety proxies using exposure indicators from `TACE.TACEDAT`, `TART.TARSDAT`, `IMU.IMUSDAT`, and `PATE.PATEYN`; calculate screening rates from `AE_HQ.AEYN` and `SAE_HQ.SAEYN`; summarize post-exposure laboratory signals from `LB_BB`, `LB_BR`, and `LB_TT`.	Report confounder imbalance using `BCLC.BCLC`, `CP.CPGRA`, and `ECOG.ECOG`; frame results as descriptive safety proxies.

Table 18: Efficacy benchmark instances.

ID	Prompt text	Required or expected evidence
E1-G	Analyze recurrence-related outcomes after surgery.	Locate surgery dates and recurrence dates in `VT`.
E1-A	Analyze recurrence-free outcomes using surgery date as time origin, recurrence events from follow-up records, and clinically relevant baseline risk strata.	Handle multiple follow-up visits and competing or missing dates.
E1-M	Estimate recurrence-free survival proxies using `SG.SGDAT` as index date; define recurrence as the earliest non-missing `VT.PADHDAT` or `VT.PADRDAT`; censor at latest `VT.VTDAT1` or `VT.VTDAT2`; stratify by `BCLC.BCLC`, `CP.CPGRA`, and exposure from `TACE`, `TART`, `IMU`, or `PATE`.	Use earliest valid recurrence date and valid time ordering; generate Kaplan–Meier curves only when dates are sufficient.
E2-G	Analyze overall survival patterns in the cohort.	Identify death dates and follow-up dates.
E2-A	Analyze overall survival by deriving time from surgery to death or last follow-up and comparing clinically meaningful strata.	Use `DEDAT1`/`DEDAT2` and `VT` follow-up dates; disclose death-date limitations.
E2-M	Estimate overall-survival proxies using `SG.SGDAT` as index date; define death date as earliest non-missing `VT.DEDAT1` or `VT.DEDAT2`; censor at latest valid `VT.VTDAT1` or `VT.VTDAT2`; compare treatment exposures and adjust or stratify descriptively by `DM.SEX`, `DM.AGE`, `BCLC.BCLC`, `CP.CPGRA`, and `ECOG.ECOG` when available.	Note that `DEDAT2` is unavailable in the current data; avoid over-adjusted models when events are limited.
E3-G	Evaluate treatment response and tumor-marker dynamics over follow-up.	Combine efficacy assessment and biomarker sheets.
E3-A	Evaluate response and tumor-marker dynamics by summarizing hospital/telephone efficacy evaluations and longitudinal AFP, CA199, and DCP trajectories by visit and exposure.	Combine categorical response information with biomarker trends.
E3-M	Analyze response and tumor-marker trends using `VT.CEVYN`, `VT.CUEV`, `VT.VTDAT1`, `VT.VTDAT2`, plus `LB_AFP.LBAFPDAT`, `AFPRES`, `CA199RES`, and `DCPRES`; align measurements by `Subject_Id` and date relative to `SG.SGDAT`; summarize trajectories by exposure to `TACE`, `TART`, or `IMU`.	If formal response criteria are not encoded, report observed categories and marker changes without RECIST claims.

Table 19: Health economics and resource-use benchmark instances.

ID	Prompt text	Required or expected evidence
H1-G	Analyze postoperative treatment resource utilization in the cohort.	Recognize that the task concerns utilization rather than direct cost.
H1-A	Analyze postoperative treatment resource utilization by quantifying use and timing of TACE, targeted therapy, immunotherapy, postoperative retreatment, and concomitant medications.	Use procedure and drug counts with dates as utilization proxies.
H1-M	Quantify treatment-utilization intensity by deriving per-subject counts and time-to-first-use for `TACE.TACEDAT`, `TART.TARSDAT`, `IMU.IMUSDAT`, `PATE.PATEYN/PATETR*`, and `CM.CMCAT/CMSTDAT`; summarize utilization distributions and stratify by `BCLC.BCLC` and recurrence status when feasible.	Explicitly state that no direct cost or billing variables are present.
H2-G	Estimate follow-up and testing burden associated with the RWS cohort.	Use visit and form/test counts as burden proxies.
H2-A	Estimate follow-up and testing burden by counting follow-up visits, efficacy assessments, laboratory panels, tumor-marker checks, coagulation tests, ECOG assessments, and QoL forms per patient.	Summarize burden by patient and visit.
H2-M	Build a follow-up resource-burden profile by counting records in `CPH_FT`, `VT`, `LB_AFP`, `LB_BB`, `LB_BR`, `LB_TT`, `ECOG`, and `SF`; compute total follow-up forms/tests, median counts per subject, and distribution by `Visit_No`; use `SG.SGDAT` and latest `VT.VTDAT1/VT.VTDAT2` to contextualize burden over time.	Frame the result as utilization burden, not economic cost.
H3-G	Explore quality-of-life evidence and its relationship to treatment burden.	Discover SF-12 sparsity and avoid overclaiming.
H3-A	Explore QoL and treatment burden using SF-12 items when available; summarize missingness and compare available QoL responses across treatment-utilization groups.	Emphasize sparse SF item completeness.
H3-M	Analyze QoL/resource-use proxy feasibility by inspecting `SF.SFYN` and `SF1`–`SF12`; compute complete-item availability; when enough records exist, summarize item-level or total scores by exposure groups from `TACE`, `TART`, `IMU`, and `PATE`; otherwise report feasibility and missingness only.	Terminate formal QoL comparison when usable SF-12 item records are too sparse.

Table 20: Public-RWS external validation datasets. These datasets test cross-dataset transfer and contract diagnostics, not additional full HCC-style longitudinal RWS validity.

Dataset	Structure	Task focus	Boundary
UCI Diabetes 130-US Hospitals	101,766 encounters, 50 columns; the department-split workbook is used only as a sheet-navigation variant.	Readmission proxy, medication process, utilization, coded variables, missingness.	Encounter-level data with limited longitudinal structure.
NHANES 2017–2018 subset	10 linked XPT/CSV tables with 3,036–19,643 rows per table, linked by `SEQN`.	Survey cohort construction, labs, medications, conditions, demographic covariates.	Cross-sectional survey data, not EHR.
TCGA-LIHC clinical metadata	377 cases, 137 clinical metadata columns.	HCC metadata, vital status, treatment fields, follow-up proxy fields.	Cancer registry-like clinical metadata, not routine-care EHR.

information, artifact templates, validator feedback, and the same repair budget, but not ProCARE's IR compiler, certificate admission gate, certified reporting DAG, or internal planner state. This appendix records the dataset boundaries, task objectives, claim policies, and prompt text needed to rerun the transfer suite. For baselines without explicit evidence certificates, CTC, UCR, and CLV are computed by post-hoc evidence reconstruction over submitted artifacts and reports.

Listing 1: Prompt set for Public-RWS-18.

```
UCI Diabetes -- Objective U1: Readmission proxy profiling
Low: Analyze the diabetes hospital encounter data and identify which patient groups appear
     to have higher readmission burden.
Medium: Using the readmitted field as a proxy outcome, summarize early-readmission patterns
     by age, admission type, discharge disposition, and prior utilization variables.
High: Using readmitted = "<30" as the early-readmission proxy outcome, compute
     subgroup-level rates by age band, admission_type_id, discharge_disposition_id,
     number_inpatient, and number_emergency. Produce a table, one figure, and a report
     restricted to descriptive or proxy-association claims with evidence certificates for
     all factual statements.
```

Table 21: Task objectives in Public-RWS-18. Each objective is evaluated under low-, medium-, and high-specificity prompts.

Dataset	Objective	Contract stressor
UCI Diabetes	Readmission proxy profiling	Outcome binding, coded variables, missingness, conservative claim scope.
UCI Diabetes	Medication and utilization analysis	Exposure/proxy roles, subgroup aggregation, artifact traceability.
NHANES	Diabetes/cardiometabolic cohort analysis	Multi-table joins by `SEQN`, lab-defined variables, survey-table grounding.
NHANES	Medication and condition association analysis	Questionnaire, laboratory, and prescription-medication grounding under a cross-sectional claim policy.
TCGA-LIHC	HCC case and vital-status profiling	Domain transfer, oncology metadata grounding, missingness disclosure.
TCGA-LIHC	Treatment/follow-up metadata analysis	Sparse fields, proxy endpoint handling, limitation reporting.

Table 22: Dataset-specific claim policies for Public-RWS-18.

Dataset	Allowed claims	Forbidden claims
UCI Diabetes	Descriptive and readmission-proxy association claims.	Causal treatment effect, safety conclusion, longitudinal endpoint claim.
NHANES	Descriptive and cross-sectional association claims.	Temporal effect, incidence claim, causal effect.
TCGA-LIHC	Descriptive oncology metadata and follow-up-proxy claims.	Treatment efficacy, causal survival effect, routine-care EHR claim.

```
UCI Diabetes -- Objective U2: Medication and utilization analysis
Low: Analyze whether medication and hospital-utilization patterns differ across diabetes
     encounters.
Medium: Summarize insulin, change, diabetesMed, A1Cresult, time_in_hospital,
     num_lab_procedures, num_medications, and num_procedures, and describe any supported
     association with readmitted.
High: Compute utilization summaries for time_in_hospital, num_lab_procedures,
     num_medications, and num_procedures stratified by insulin, change, and diabetesMed.
     Compare early-readmission proxy rates across medication-process groups, and avoid
     causal treatment-effect language.

NHANES -- Objective N1: Diabetes/cardiometabolic cohort analysis
Low: Use the NHANES tables to profile diabetes and cardiometabolic risk patterns in the
     population sample.
Medium: Join demographics, examination, blood pressure, diabetes questionnaire, laboratory,
     and medical-condition tables by SEQN. Summarize diabetes-related variables and
     cardiometabolic measurements by age and sex.
High: Construct a cross-sectional NHANES cohort using SEQN-linked tables. Define
     diabetes-related status using supported questionnaire or laboratory fields, summarize
     blood pressure and relevant laboratory measures by demographic subgroup, and report
     only descriptive or cross-sectional association claims.

NHANES -- Objective N2: Medication and condition association analysis
Low: Analyze how prescription medication records relate to self-reported medical conditions
     in NHANES.
Medium: Join prescription medication, medical conditions, demographics, and laboratory
     tables by SEQN. Summarize medication exposure groups and compare supported condition or
     laboratory profiles.
High: Using SEQN-linked medication and condition tables, construct medication-exposure
     groups when supported by the codebooks, summarize condition prevalence and laboratory
     measurements, and produce a report that explicitly states cross-sectional design
     limitations.
```

```
TCGA-LIHC -- Objective T1: HCC case and vital-status profiling
Low: Profile the TCGA-LIHC clinical metadata and summarize major patient, diagnosis, and
     vital-status patterns.
Medium: Summarize demographics, diagnosis-related fields, vital status, and available
     follow-up-related fields in TCGA-LIHC. Identify missingness and fields that can support
     survival-proxy reporting.
High: Using TCGA-LIHC clinical metadata, produce a case-profile table covering age, sex,
     diagnosis-related fields, vital status, days-to-death or follow-up-related fields when
     available, and missingness. Report only descriptive oncology-metadata claims with
     evidence certificates.

TCGA-LIHC -- Objective T2: Treatment/follow-up metadata analysis
Low: Analyze whether available treatment and follow-up metadata in TCGA-LIHC support
     subgroup comparisons.
Medium: Identify treatment-related, diagnosis-related, and follow-up-related fields in
     TCGA-LIHC, assess their completeness, and perform only comparisons supported by valid
     profile entries.
High: Using TCGA-LIHC clinical metadata, identify valid treatment-related and
     follow-up-related fields, summarize completeness by subgroup, and generate a
     conservative report that distinguishes observed metadata patterns from treatment-effect
     or causal claims.
```

B.7 Worksheet inventory

Table 23: Standardized worksheet inventory in the Excel dataset.

Prefix	#Fields	#Rows	Sheet Description	Differential field examples
SYSVAR	2	27	System variables and metadata.	Variable, Description
DATA_RANGE	2	4	Data range / center mapping and allocation notes.	Center, RangeCode
ECRF	8	54	eCRF form catalog and design attributes.	FormCode, FormName, FormType
CPH_FT	22	11614	Header/footer records for visit-/form-level.	VISTNM, SITNO, SUBNO
CO	21	1296	Study cover page and subject identification fields.	SITENO, SUBJNO, SUBINI
DS	19	1296	Informed consent date and related tracking fields.	DSDAT
DM	27	1295	Demographics and baseline characteristics.	SEX, BRTHDAT, AGE
PH	123	1262	Medical history and prior conditions.	PHSMYN, SMNUM, SMYER
PH1	123	1	Medical history (continuation / overflow record).	PH1SMYN, SMNUM1, SMYER1
HBCV	49	4282	HBV/HCV-related laboratory assessments.	HBVYN, HBVDAT, HBSAGRES
HBCV1	49	1	HBV/HCV laboratory assessments.	HBV1YN, HBV1DAT, HBSAGRE1
VS	25	1233	Vital signs measurements.	VSRES, VSTEMP, VSSBP
CP	32	1262	Child-Pugh assessment and component scoring.	CPYN, CPRES, CPGRA
IE	31	549	Eligibility criteria assessment.	IEDAT, INCL1, INCL2
IE1	30	714	Eligibility criteria (alternate set)..	INCL01, INCL02, EXCL01
TUBA	94	1295	Intraoperative tumor pathology / surgical findings.	TUBADSC, TUBATYPE, TUBAGRA
BCLC	28	1261	BCLC staging for primary liver cancer.	BCLC, PLICA, PLICA1
BCLC1	28	1	BCLC staging (continuation record).	BCLC1, BCPLICA, BCPLICA1
VT	96	5321	Efficacy assessment interview and follow-up.	VTTYP, VTDAT1, VTDAT2
VT1	96	2	Efficacy assessment (continuation record).	VT1TYP, VT1DAT1, VT1DAT2
LB_AFP	57	4613	Tumor marker laboratory tests.	AFPYN, LBAFPDAT, AFPRES
AFP2	26	4166	Other tumor marker laboratory tests.	AFP2DAT, AFP2NM, AFP2RES
LB_CD	87	4392	Lymphocyte subset testing panel.	CDYN, CDDAT, CD3PRES
LB_BR	93	4474	Complete blood count (CBC).	BRYN, BRDAT, WBCRES

Table 23: Standardized worksheet inventory in the Excel dataset (continued).

Prefix	#Fields	#Rows	Sheet Description	Differential field examples
LB_BB	123	4583	Serum biochemistry and hepatic function panel.	LBBBYN, LBBBDAT, ALTRES
LB_BB1	105	1	Serum biochemistry.	LBBB1YN, LBBB1DAT, ALTRES1
LB_CYT	75	4325	Cytokine testing panel.	LBCYTYN, LBCYTDAT, IL1RES
LB_TT	55	4466	Coagulation function testing panel.	LBTTYN, LBTTDAT, PTRES
ECOG	22	4459	ECOG performance status assessment.	ECOGYN, ECOG
PATE	39	3265	Postoperative retreatment information.	PATEYN, PATETR1, PATETR2
SG	21	1325	Surgery-related records and key attributes.	SGDAT, SGTYP
RFT	75	1323	Combined radiofrequency therapy records.	UEXYN, UEXLO1, UEXLO2
FIN	20	1325	Number of lesions and summary lesion count.	FINM
FI	74	1489	Lesion-level details (location and characteristics).	FILO1, FILO2, FILO3
TRDAT	20	27	Postoperative treatment start date/time fields.	TRDAT, TRDATS
THET_HQ	23	123	Chemotherapy screening / question page.	THETYN, THETDSC, THETUK
THET	30	154	Chemotherapy administration records.	THETPHA, THESDAT, THETONGO
TART_HQ	20	453	Targeted therapy screening / question page.	TARTYN, TARTYN_DEC
TART	35	303	Targeted therapy administration records.	TARTPHA, TARSDAT, TARTONGO
IMU_HQ	20	404	Immunotherapy screening / question page.	IMUYN, IMUYN_DEC
IMU	35	290	Immunotherapy administration records.	IMUPHA, IMUSDAT, IMUONGO
TACE_HQ	20	615	TACE screening / question page.	TACEYN, TACEYN_DEC
TACE	30	609	TACE procedure and regimen details.	TACEDAT, TACETYP1, TACETYP2
CMED	20	283	Traditional Chinese medicine treatment records.	CMEDYN, CMEDYN_DEC
RADI_HQ	52	15	Radiotherapy screening / question page.	RADIYN, RADIMD1, RADIMD2
RADI	21	6	Radiotherapy administration records.	RADIGN, RADIDAT, RADIDSC
SF	44	3183	SF-12 quality-of-life questionnaire.	SFYN, SF1, SF2
CM_HQ	20	1235	Concomitant medication screening / question.	CMYN, CMYN_DEC
CM	34	6632	Concomitant medication records.	CMCAT, CMINDC, CMDOSE
LB_BP	24	1232	Blood typing and related laboratory annotations.	LBBPYN, LBBPDAT, LBBPSIG
AE_HQ	20	1231	Adverse event screening / question page.	AEYN, AEYN_DEC
AE	34	1	Adverse event records.	AETERM, AESTDAT, AEENDAT
SAE	74	1	Serious adverse event records and site contact.	SAECAT1, SAECAT2, SAERPDAT
SAE_HQ	20	1231	Serious adverse event screening / question page.	SAEYN, SAEYN_DEC
LOAB_HQ	20	51	Local ablation screening / question page.	LOABYN, LOABYN_DEC
LOAB	21	51	Local ablation procedure records.	LOABDAT, LOABMEAR, LOABMEAR_DEC

C Method and Contract Implementation Details

This section records the implementation artifacts used by the profiling, RWS-IR compilation, violation-guided execution, and certified reporting stages.

C.1 Profile schema Φ

The main paper defines the four profile blocks. The implementation stores the same object as a JSON contract with stable keys for high-level structural summaries, high-level knowledge roles, field-level

structural statistics, and field-level knowledge mappings. Downstream stages do not consume free-form profile prose; they read these keys directly when compiling RWS-IR fields, validating generated code, and admitting report claims.

Structural profile $\Phi_{\cdot,S}$. For each table/sheet and column, we record deterministic statistics, including: inferred type, missingness, cardinality, value ranges, common categories, and candidate join keys.

Knowledge profile $\Phi_{\cdot,K}$. We map raw fields to standardized biomedical concepts (e.g., UMLS/SNOMED CT/ICD/LOINC) by candidate generation followed by deterministic validation checks: (i) *semantic type compatibility*, and (ii) *value-range consistency*. Unmappable or ambiguous fields are retained with flags to propagate uncertainty into RWS-IR compilation, execution, and reporting.

Implementation serialization and fallback. The implementation serializes Φ as a JSON object with four named blocks: `high_structural`, `low_structural`, `high_knowledge`, and `low_knowledge`. The high-structural block stores dataset-level table relationships, candidate subject and visit keys, and worksheet-level roles. The low-structural block stores field-level type inference, missingness, cardinality, range checks, common categorical values, date-parsing status, and candidate temporal anchors. The high-knowledge block stores study-level roles, including index-date schema, exposure, outcome, censoring, confounder, proxy, and claim-scope roles. The low-knowledge block stores ontology candidates, accepted mappings, validation status, units, value-range plausibility, temporal-role compatibility, and ambiguity or rejection causes.

For each biomedical mapping candidate, the profiler applies semantic-type compatibility, unit consistency, value-range plausibility, and temporal-role compatibility checks. A field that fails these checks is retained in $\Phi_{L,K}$ with a failure tag rather than removed. Such fields remain available for structure-grounded summaries, missingness reporting, and data-quality diagnostics, but cannot authorize clinical-role binding, endpoint derivation, safety conclusions, causal claims, or high-stakes report language unless an explicit proxy assumption is recorded in the RWS-IR and admitted by the claim policy.

C.2 Concrete heterogeneous profile instance

Listing 2 shows a compact de-identified JSON instance of Φ from the HCC benchmark. The example follows the same schema used by the implementation and contains aggregate profile statistics and field-level metadata; record-level values are provided in the released de-identified HCC tables rather than repeated in the text.

Listing 2: Compact JSON instance of the heterogeneous data profile Φ.

```
{
  "serialization": "JSON",
  "notation_map": {
    "Phi^{H,S}": "heterogeneous_profile.high_structural",
    "Phi^{L,S}": "heterogeneous_profile.low_structural",
    "Phi^{H,K}": "heterogeneous_profile.high_knowledge",
    "Phi^{L,K}": "heterogeneous_profile.low_knowledge"
  },
  "heterogeneous_profile": {
    "high_structural": {
      "dataset_summary": {
        "workbook_type": "multi-sheet eCRF workbook",
        "benchmark_subject_count": 1295,
        "benchmark_visit_count": 12847
      },
      "sheet_inventory_subset": [
        {
          "sheet": "DM--Demographics",
          "role": "baseline demographics",
          "estimated_data_rows": 1295,
          "columns": 27,
          "join_keys": ["Subject_Id"]
        },
        {
          "sheet": "SG--Surgery",
          "role": "index surgery",
          "estimated_data_rows": 1324,
          "columns": 21,
```

```
          "join_keys": ["Subject_Id"]
        },
        {
          "sheet": "VT--Follow-up",
          "role": "postoperative follow-up",
          "estimated_data_rows": 5320,
          "columns": 96,
          "join_keys": ["Subject_Id", "Visit_No"]
        }
      ],
      "relationship_hints": [
        "DM.Subject_Id joins SG.Subject_Id and VT.Subject_Id.",
        "SG.SGDAT is the preferred index date.",
        "VT.Visit_No encodes repeated follow-up windows."
      ]
    },
    "low_structural": {
      "field_profiles": {
        "DM.SEX_DEC": {
          "dtype": "categorical",
          "missing_rate": 0.0,
          "top_values": {"male": 165, "female": 34}
        },
        "SG.SGDAT": {
          "dtype": "date-like text",
          "missing_rate": 0.005,
          "semantic_hint": "index surgery date"
        },
        "VT.PADHDAT": {
          "dtype": "partial date-like categorical",
          "missing_rate": 0.915,
          "quality_note": "sparse recurrence date"
        },
        "VT.DEDAT2": {
          "dtype": "date-like categorical",
          "missing_rate": 0.995,
          "quality_note": "nearly unavailable"
        }
      }
    },
    "high_knowledge": {
      "variable_roles": {
        "id": ["DM.Subject_Id", "SG.Subject_Id", "VT.Subject_Id"],
        "time_anchor": ["SG.SGDAT"],
        "baseline_covariate": [
          "DM.AGE", "DM.SEX_DEC", "BCLC.BCLC_DEC",
          "CP.CPGRA", "ECOG.ECOG"
        ],
        "outcome": [
          "VT.PADHDAT", "VT.PADRDAT",
          "VT.DEDAT1", "VT.DEDAT2"
        ]
      },
      "feasibility_assessment": "Cohort and follow-up summaries are feasible; survival
          analyses require an availability check."
    },
    "low_knowledge": {
      "ontology_mapping": {
        "DM.SEX_DEC": "Biological sex",
        "DM.AGE": "Age",
        "SG.SGDAT": "Surgical procedure date",
        "VT.PADHDAT": "Recurrence date"
      },
      "validation_status": {
        "DM.HEIGHT": "ambiguous: numeric values mixed with unknown tokens",
        "VT.PADHDAT": "valid_but_sparse",
        "VT.DEDAT2": "invalid_for_primary_survival_endpoint"
      },
      "analysis_constraints": [
        "Use only fields listed in field_profiles.",
        "Treat unknown or partial date tokens as missing unless the method supports
            interval censoring.",
        "Do not infer recurrence or death outcomes from missing date fields."
      ]
    },
    "ambiguity_flags": [
      "DM.HEIGHT", "DM.WEIGHT",
      "VT.PADHDAT", "VT.DEDAT2"
    ]
  }
}
```

Table 24: Machine-readable fields in the RWS-IR and static checks before execution.

Symbol	Serialized key	Contents	Static admissibility checks
D	`study_design`	Study type, estimand boundary, and whether the requested output is descriptive, associative, or inferential.	Must be compatible with available tables, observed design, and claim policy.
G	`analysis_grain`	Subject-, visit-, encounter-, field-, or record-level analysis unit.	Must match key multiplicities and downstream aggregation.
J	`join_graph`	Tables, join keys, cardinalities, and permitted join paths.	All tables and keys must exist in $\Phi_{L,S}$; declared cardinality must match profiled relationships.
K	`entity_keys`	Subject, visit, encounter, treatment, and artifact identifiers.	Required keys must be present and nondegenerate under the declared grain.
T	`temporal_anchors`	Index date, event dates, censoring dates, windows, and ordering rules.	Date fields must be parseable or declared as partial/proxy dates; impossible orderings are disallowed.
V	`variable_roles`	Exposure, outcome, censoring, covariate, subgroup, proxy, and reporting variables.	Clinical roles must be supported by $\Phi_{L,K}$, a validated proxy flag, or a recorded assumption.
M	`method`	Statistical method, fallback method, denominators, missingness policy, and diagnostics.	Method must be compatible with design, grain, variable support, and minimum data requirements.
A	`required_artifacts`	Required result tables, figures, manifests, logs, and report sections.	Artifacts must be executable in the sandbox and verifiable by manifests.
Π_q	`claim_policy`	Allowed and forbidden claim types, language constraints, and support levels.	Claim policy must be no stronger than the design and evidence support.

Table 25: Static violation types emitted by `IRCheckstatic`.

Violation type	Trigger	Allowed repair
Missing schema reference	A table, field, or key in Z_0 is absent from $\Phi_{L,S}$.	Replace with a profiled field, remove the variable, or declare infeasibility for GOAL+METHOD.
Unsupported semantic role	Exposure, endpoint, censoring, or confounder role is not validated in $\Phi_{L,K}$.	Use a validated proxy with limitation, downgrade to descriptive support, or stop.
Invalid join or grain	Join cardinality or aggregation conflicts with the declared analysis grain.	Change grain, revise the join graph, or aggregate before merging.
Invalid temporal anchor	Index, event, censoring, or follow-up dates are missing, partial, nonparseable, or ordered incorrectly.	Use a supported anchor, restrict to analyzable records, or report feasibility only.
Method infeasibility	The requested method requires unsupported design assumptions, insufficient events, or unavailable covariates.	Switch to a supported fallback method or emit an infeasibility statement.
Claim-policy excess	Π_q allows causal, safety-critical, economic-cost, or generalization claims unsupported by the data.	Restrict Π_q to descriptive, feasibility, or proxy-association claims.

C.3 RWS-IR schema and static checks

The RWS-IR is the typed study object validated before code generation. It records the study design, analysis grain, join graph, entity keys, temporal anchors, variable roles, methods, required artifacts, and claim policy. Table 24 lists the machine-readable fields and the admission checks applied before execution.

Given a candidate Z_0, `IRCheckstatic` returns an accepted RWS-IR Z and a set of static violations U_{static}. A hard violation is any reference to a missing table, missing field, invalid key, unsupported clinical role, incompatible analysis grain, impossible temporal anchor, method that requires unavailable variables, or claim policy exceeding the design. A soft violation is a downgradeable limitation, such as sparse endpoint support, partial date availability, missing units, or a proxy variable that can support descriptive but not causal claims. Only $U_{\text{static}} = \emptyset$ enters the execution workspace. When hard violations remain, ProCARE either repairs the RWS-IR, emits an infeasibility statement, or downgrades the requested analysis according to prompt specificity.

C.4 RWS-IR and evidence certificate examples

Listing 3 gives a compact RWS-IR instance for a recurrence-free survival task. The example shows the fields used by the static checker: study design, analysis grain, join graph, temporal anchors, variable-role bindings, method assumptions, required artifacts, and claim policy.

Listing 3: Compact RWS-IR example for a recurrence-free survival task.

```
{
  "study_design": "retrospective_cohort",
  "analysis_grain": "subject",
  "population": {
    "source_tables": ["DM", "SG", "VT"],
    "subject_key": "Subject_Id",
    "inclusion": ["non_missing(DM.Subject_Id)", "surgery_record_exists(SG)"]
  },
  "join_graph": [
    {"left": "DM.Subject_Id", "right": "SG.Subject_Id", "type": "one_to_one"},
    {"left": "DM.Subject_Id", "right": "VT.Subject_Id", "type": "one_to_many"}
  ],
  "temporal_anchors": {
    "index_date": "SG.SGDAT",
    "event_date_candidates": ["VT.PADHDAT", "VT.PADRDAT"],
    "censoring_date_candidates": ["VT.VTDAT1", "VT.VTDAT2"]
  },
  "variable_roles": {
    "exposure": "surgery cohort",
    "outcome": "recurrence-free survival proxy",
    "covariates": ["DM.AGE", "DM.SEX_DEC", "BCLC.BCLC_DEC", "CP.CPGRA"]
  },
  "method": {
    "primary": "Kaplan-Meier summary if temporal checks pass",
    "fallback": "descriptive follow-up availability when event dates are sparse"
  },
  "required_artifacts": ["cohort_flow", "followup_summary", "
      km_curve_or_feasibility_table"],
  "claim_policy": {
    "allowed": ["descriptive", "association_with_limitation"],
    "blocked": ["causal_effect", "safety_detail", "submission_grade_claim"]
  }
}
```

Listing 4 gives a representative evidence certificate. The certificate records the claim text, its support level, the validated artifacts and code region that produced it, the RWS-IR fields that authorize the claim, and the profile entries or assumptions used by the admission rule.

Listing 4: Evidence certificate example for an admitted factual claim.

```
{
  "claim_id": "q_rfs_03",
  "claim_text": "Recurrence-free survival can be summarized for subjects with valid
      surgery and follow-up dates.",
  "claim_type": "descriptive_feasibility",
  "support_level": "profile_and_artifact_supported",
  "trace": {
    "artifact": "results/followup_summary.json",
    "figure": "figures/km_or_feasibility.pdf",
    "code_region": "analysis.py::build_rfs_dataset",
    "rws_ir_fields": [
      "temporal_anchors.index_date",
      "temporal_anchors.event_date_candidates",
      "method.primary",
      "claim_policy.allowed"
    ],
    "profile_entries": ["SG.SGDAT", "VT.PADHDAT", "VT.PADRDAT", "VT.VTDAT1", "VT.VTDAT2"]
        ,
    "assumptions": ["partial or unknown date tokens are treated as missing"]
  },
  "admission": {
    "trace_check": true,
    "policy_check": true,
    "language_check": true
  }
}
```

C.5 Execution validators and violation-guided repair

Execution-time checks are applied after code generation because row-level temporal ordering, manifest completeness, figure creation, and artifact consistency cannot be fully resolved statically. Validators return a localized violation set $U_t = V_{\text{exec}}(Z_t, C_t, O_t, E_t, \Phi, S_t)$. Each violation records the failed contract clause, offending code or artifact, nearest valid profile entries, severity, and permitted repair actions. Listing 5 shows the JSON schema used for one violation record.

Table 26: Execution validators used before accepting an artifact bundle.

Validator	Check	Acceptance criterion
Entry-point validator	Runs `analysis.py` or `main.py` in the sandbox.	Exit code is zero and no failure sentinel such as `TASK_FAILED` appears.
Schema validator	Parses code and generated outputs for table, field, and key usage.	Every referenced table, field, join key, and output column is traceable to $\Phi_{L,S}$ or a derived artifact.
Grain and join validator	Checks duplicate inflation, one-to-many joins, and final denominator consistency.	Output denominators match the declared analysis grain or disclose aggregation rules.
Temporal validator	Recomputes index, event, censoring, and follow-up windows.	Invalid date strings and impossible orderings are filtered, logged, or reported as limitations.
Method validator	Checks cohort filters, variables, denominators, covariates, and statistical choices against $Z.M$.	Method matches the admitted RWS-IR or a declared fallback.
Artifact validator	Checks result and figure manifests, file existence, and generated figure formats.	Required tables, figures, manifests, and logs exist and are nonempty when applicable.
Claim-preparation validator	Builds the evidence map used by the report generator.	Each reportable statistic has a source artifact, producing code region, and profile/IR trace.

Algorithm 1 Execution acceptance and localized repair

Require: Accepted RWS-IR Z, profile Φ, workspace policy Π, maximum attempts B
Ensure: Accepted code C^*, artifacts O, evidence E, assumptions A, or failure trace

for $t = 1, \ldots, B$ **do**
  Generate or edit code C_t conditioned on Z, Φ, Π, and prior violations U_{t-1}.
  Execute the selected entry point in the sandbox and collect logs, result manifests, and figure manifests.
  Compute $U_t = V_{\text{exec}}(Z, C_t, O_t, E_t, \Phi, S_t)$.
  if $U_t = \emptyset$ **then**
    Accept $C^* = C_t$ and return (C^*, O_t, E_t, A_t).
  else
    Write U_t to `violation_set.json` with localized repair actions.
  end if
end for
Return the final failure trace or a supported downgrade with explicit limitations.

Listing 5: Schema for one execution-time contract violation.

```
{
  "violation_id": "u_017",
  "stage": "execution",
  "contract_clause": "temporal_anchor.index_date_before_event",
  "severity": "hard",
  "offending_artifact": "analysis.py",
  "code_region": "build_survival_dataset",
  "observed_failure": "event_date precedes SG.SGDAT for at least one record",
  "nearest_profile_entries": ["SG.SGDAT", "VT.PADHDAT", "VT.PADRDAT"],
  "permitted_actions": ["filter_invalid_temporal_pairs", "report_temporal_limitation"],
  "blocks_acceptance": true
}
```

C.6 Evidence certificates and claim admission

A factual claim is any report sentence that asserts a dataset property, cohort count, variable availability, statistical result, visualization interpretation, clinical association, feasibility conclusion, or limitation derived from the data. Section headings, procedural statements, and generic background statements are not counted unless they contain dataset-specific factual content. For ProCARE, certificates are emitted during report generation. For baselines without explicit certificates, the same checker performs post-hoc reconstruction from the submitted code, logs, artifacts, and report.

Each claim q receives a certificate $e_q = \langle o_q, c_q, z_q, \phi_q, a_q \rangle$, where o_q is a supporting artifact, c_q is the executed code region that created the artifact, z_q is the RWS-IR field or method authorizing the claim, ϕ_q is the relevant profile entry, and a_q is an optional recorded assumption or limitation. The trace check passes only when all required links exist and agree on variables, denominators, analysis grain, temporal windows, and method. The policy check passes only when the claim type and wording are allowed by $Z.\Pi_q$ at the available support level.

Table 27: Claim types used by the certificate gate.

Claim type	Allowed support	Examples of blocked wording
Descriptive	Validated table, count, summary statistic, figure, or profile entry.	Claims that imply generalization beyond the analyzed cohort.
Feasibility or data-quality	Profile entries, missingness summaries, temporal-validity logs, or execution artifacts.	Treating missingness or feasibility as clinical outcome evidence.
Proxy association	Valid proxy flag, artifact support, and explicit limitation.	Causal treatment-effect language or safety conclusion language.
Causal or comparative effect	Requires design assumptions, adjustment plan, and method support declared in Z.	Causal language when only descriptive or proxy support is available.
Safety-critical or economic-cost	Requires detailed safety records or direct cost/billing fields.	Detailed AE/SAE term claims when only screening pages exist; cost-effectiveness claims without cost variables.

Table 28: Operational mapping between profile-grounded state roles and implementation artifacts.

State role	Stored artifacts	Example contents	Accessed by
Execution trace	`exec_logs.json`, `exec_status.json`, `opencode_live.log`, `violation_set.json`, audit records.	execution traces, contract violations, validation outcomes, disclosed assumptions.	violation-guided execution, report generator, frontend status view.
Working context	`excel_info.json`, `rws_ir.json`, `execution_spec.md`, selected sheet payloads.	selected sheets, required fields, join keys, variable roles, temporal anchors, method specification, claim policy.	IR compiler, execution workspace, report context builder.
Profile semantics	$\Phi^{H,K}$, $\Phi^{L,K}$, ontology mappings, reporting prompts.	exposure/outcome/covariate roles, plausibility checks, clinical terminology, section semantics.	IR compiler, code generator, report DAG nodes.
Certified outputs	result manifests, figure manifests, evidence certificates, report sections, disclosed assumptions.	validated tables and figures, claim traces, support levels, limitations, section drafts.	claim checker, report DAG, final report renderer.
Operational policy	IR compilation, profiling, coding, figure, validation, and report-generation policies.	coding rules, artifact schemas, figure manifest schema, section templates, validation gates, claim policies.	execution workspace, report builder.

Let N_q be the number of extracted factual claims, C_q the number with admitted evidence certificates, U_q the number without admissible evidence, and L_q the number whose language exceeds the claim policy. We compute

$$\mathrm{CTC} = C_q/N_q, \qquad \mathrm{UCR} = U_q/N_q, \qquad \mathrm{CLV} = L_q/N_q.$$

A claim can be both unsupported and a language violation, but each metric is normalized by the same factual-claim denominator. For the claim-diagnostic table in the main paper, metrics are computed per run and then averaged across the 12 tasks in each prompt-specificity group; the overall row averages across all 36 HCC tasks.

C.7 Profile-grounded state artifacts

The workflow state is represented by typed JSON artifacts, workspace files, skills, and validation logs. No vector database is used. Each downstream module reads the relevant state artifact from named files or typed prompt payloads, so intermediate decisions remain auditable without re-querying raw tables. Concretely, the persistent dataset state is the cached profile Φ, while the per-job state $\mathcal{S}_t$ may contain `excel_info.json`, `rws_ir.json`, `violation_set.json`, `execution_spec.md`, generated code, execution logs, result manifests, figure manifests, evidence certificates, report sections, and disclosed assumptions. These files are bookkeeping carriers for the profile contract: downstream stages may reuse a fact only when it can be traced to profiled tables, validated fields, executable artifacts, or explicitly recorded assumptions.

C.8 Ontology mapping coverage

The knowledge profiler maps raw fields to biomedical concepts from UMLS, SNOMED CT, ICD-10, and LOINC through candidate generation followed by type-compatibility and value-range checks. Overall field-level mapping coverage is 0.83. Coverage for critical task fields, including exposures, outcomes, confounders, and temporal anchors, is 0.91.

Algorithm 2 Profile-grounded state update and access in ProCARE

Require: User query q, dataset D, profile cache $\mathcal{C}$
Ensure: Analysis artifacts, report, and updated job state
1: Compute dataset hash $h = \mathrm{MD5}(D)$.
2: Load Φ from $\mathcal{C}[h]$ if available; otherwise profile D and cache Φ.
3: Create job workspace W and write the profile into `excel_info.json`.
4: Compile candidate RWS-IR Z_0 from q and the high-level profile.
5: Run static IR checks against Φ and the job state.
6: Write accepted Z to `rws_ir.json` and write execution context to `execution_spec.md`.
7: Run violation-guided execution using Z, low-level profile blocks, and operational policy artifacts.
8: Append execution logs, violation sets, validation results, and figure manifests to the job state.
9: Generate certified report sections from the reporting DAG using Φ, Z, logs, figures, evidence certificates, and existing sections.

Table 29: Required artifact-bundle schema.

Artifact	Typical file name	Evaluation use
RWS-IR	`rws_ir.json`	PF, method validation, temporal and grain checks, claim policy.
Executable entry point	`analysis.py` or `main.py`	XE, CQ, artifact generation, code-region traces.
Execution logs	`exec_logs.json`, `exec_status.json`	Runtime pass/fail, violation localization, evidence reconstruction.
Results manifest	`results_manifest.json`	Result-table existence, denominators, generated statistics.
Figure manifest	`figure_manifest.json`	Figure existence, caption/axis metadata, chart-fidelity checks.
Violation set	`violation_set.json`	VRR and repair diagnostics.
Evidence certificates	`evidence_certificates.json`	CTC, UCR, CLV, report traceability.
Assumptions and limitations	`assumptions.json` or report section	Downgrade handling and policy-compatible language checks.
Final report	`report.md`, `report.json`, or rendered report	RS, SL, factual-claim extraction, evidence-policy matching.

Unmapped fields remain in $\Phi_{L,K}$ with failure tags. The main failure categories are abbreviations or local jargon without a stable ontology match (0.07), type incompatibility between candidate concepts and observed values (0.05), ambiguity among indistinguishable candidates (0.03), and mixed units or inconsistent coding systems (0.02).

C.9 Artifact bundle contract

The prompt schema contains the task definition, optional guidance, and required deliverables. Each run returns a validated RWS-IR with required tables, join keys, variables, temporal anchors, methods, artifacts, and claim policy; one executable entry point (`analysis.py` or `main.py`) with optional helper modules for data loading and artifact generation; and a structured scientific report with evidence certificates. Generated SQL or pandas operations may reference only tables and columns present in the profile.

The contract depends on prompt specificity. Under `GOAL+METHOD`, the agent follows the requested variables and stops when required data are unavailable. Under `GOAL` and `GOAL+APPROACH`, the agent may generate the closest feasible analysis, but the report must disclose data limitations and method deviations.

The artifact bundle is the unit of evaluation. A bundle is accepted for full scoring only if it contains the files in Table 29. Missing execution-dependent artifacts produce N/A for metrics that cannot be computed, while schema, report, and claim diagnostics are still computed when the submitted artifacts allow post-hoc checking.

C.10 Profile-constrained coding harness

The code-generation stage is implemented as a controlled execution workspace. The workspace writes the profile, RWS-IR, local policies, generated code, figures, logs, violation sets, and validation records into one job directory. The coding agent can edit and execute files inside this workspace, while the backend reruns the selected entry point and rejects incomplete artifact bundles. Table 30 maps the execution components to concrete implementation artifacts.

Table 30: Concrete artifacts used by the profile-constrained coding harness.

Harness component	Concrete artifact	Purpose
Workspace W	Isolated job directory with generated scripts, plots, logs, and report fragments.	Keeps one RWS task reproducible and separates generated files from source code.
Context X	`rws_ir.json`, `excel_info.json`, and `execution_spec.md`.	Supplies objective, selected sheets, required fields, join keys, temporal anchors, method choices, and claim policy.
Policies Π	Local coding instructions, figure policy, `publication_plotting.py`, and artifact schema.	Converts profile and reporting requirements into file-level rules that the coding agent must follow.
Tools $\mathcal{T}$	File editing, Python execution, local skill use, and sandbox validation.	Allows edit-run-fix behavior inside the controlled workspace.
Validators $\mathcal{V}$	Entry-point execution, `TASK_FAILED` sentinel check, `figure_manifest.json`, and PNG/PDF/SVG existence checks.	Turns soft generation instructions into completion gates for executable artifacts.
Traces L	`opencode_live.log`, `exec_logs.json`, `exec_status.json`, and `violation_set.json`.	Records evidence for violation-guided repair, failure reporting, and later report grounding.

Table 31: Task metadata used in the reporting DAG.

Symbol	Meaning
$\mathrm{prod}(t)$	Artifacts produced by task t, such as tables, figures, statistics, or text.
$\mathrm{req}(t)$	Prerequisites used by task t, such as variables, cohorts, temporal anchors, or method assumptions.

The constraint intent remains the same, but enforcement happens through observable gates instead of a separate symbolic solver. Schema constraints are checked against `excel_info.json`; methodological constraints are checked against the RWS-IR and profile semantics; artifact constraints are checked by execution and manifest validation. If the profile does not support a required variable, temporal anchor, or statistical procedure, the execution workspace requires a failed run or an explicit limitation rather than an unsupported substitute.

C.11 DAG reporting dependency rules

Report generation uses a dependency DAG $G = (T, E)$. Each node $t \in T$ is an atomic reporting task that either constructs a validated evidence artifact or renders a report section from validated artifacts. Each directed edge $(t_i, t_j) \in E$ indicates that t_j requires artifacts produced by t_i. Table 31 defines the task metadata used by the graph.

Required chapter schema for RS. The minimal required chapters are listed in Appendix C.13. RS is computed as the fraction of required chapters covered with non-trivial content.

Support score and edge weights. For each dependency edge (t_i, t_j), we assign a weight:

$$w_{ij} = s_{ij} \cdot \sigma(0.0 + 3.0\, c_{ij})\,, \tag{6}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ maps to $[0, 1]$ and $s_{ij} \in \{0, 1\}$ is a feasibility gate indicating whether t_j can run from the available artifacts and recorded assumptions.

We compute c_{ij} from the four deterministic signals in Table 32:

$$c_{ij} = 0.35 S_{\mathrm{var}} + 0.25 S_{\mathrm{temp}} + 0.25 S_{\mathrm{size}} + 0.15 S_{\mathrm{cons}}. \tag{7}$$

All experiments use $\alpha = 0.0$, $\beta = 3.0$, $w_{\mathrm{var}} = 0.35$, $w_{\mathrm{temp}} = 0.25$, $w_{\mathrm{size}} = 0.25$, $w_{\mathrm{cons}} = 0.15$, and $N_{\min} = 30$. These constants are fixed before evaluation. They encode the ordering preference that validated variable support, temporal validity, effective sample size, and method-design consistency should be present before a result section is promoted above a limitation or feasibility section. The traversal rule changes report ordering and section expansion only; it does not change executed statistics or certify unsupported claims.

Table 32: Support signals for reporting-task prioritization.

Signal	Definition
S_{var}	Variable availability in the analysis-ready cohort, with missingness penalties. One instantiation is $\frac{1}{\lvert\mathcal{V}\rvert}\sum_{v\in\mathcal{V}}\max(0,1-\text{miss}(v))$.
S_{temp}	Temporal validity, defined as $1-r_{\text{invalid}}$, where r_{invalid} is the fraction of missing, invalid, or impossible temporal prerequisites.
S_{size}	Sample adequacy, defined as $\min(1, N_{\text{eff}}/30)$ after filtering and joins.
S_{cons}	Method-design consistency, with partial credit only when execution is feasible under recorded assumptions.

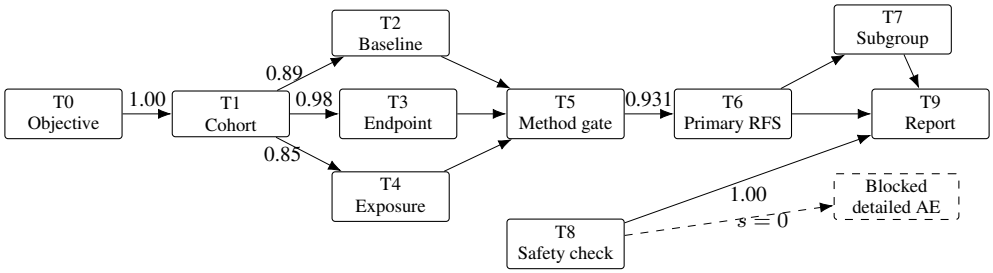


Figure 3: Instantiated reporting DAG for the recurrence-free survival task. Solid edges denote feasible artifact dependencies and show selected support scores; the dashed edge records the blocked AE-detail analysis, which is routed to limitations rather than promoted as a safety result.

Ordering policy. We generate report outlines with a weighted topological traversal. At each step, among tasks whose prerequisites are satisfied, we select:

$$t^\star = \arg\max_{t\in\text{frontier}} \sum_{(u,t)\in E} w_{ut}. \tag{8}$$

C.12 Instantiated reporting DAG

For the recurrence-free survival benchmark instance, the task uses `SG.SGDAT` as the index date, the earliest non-missing `VT.PADHDAT`/`VT.PADRDAT` as the recurrence event date, and the latest non-missing `VT.VTDAT1`/`VT.VTDAT2` as censoring time. In the workbook, the surgery cohort contains 1,295 subjects, 1,215 subjects have analyzable RFS follow-up, and 304 subjects have valid recurrence events. The same graph shows evidence gating: because the `AE` and `SAE` detail tables contain no valid records, detailed AE profiling is blocked and the evidence gap is propagated to the limitations section.

C.13 Standard report schema

The reporting engine targets the sections in Table 35.

D Experimental Protocol and Reproducibility

D.1 Execution environment and cost accounting

Generated analysis code is rerun from a clean job workspace in a Python 3.9 sandbox with network access disabled. The execution image includes `pandas`, `numpy`, `scipy`, `matplotlib`, `seaborn`, and `statsmodels`. The evaluator records stdout, stderr, return code, generated files, result manifests, figure manifests, violation sets, and failure sentinels. LLM calls are made outside the code-execution sandbox; generated code cannot fetch external data during scoring.

Token use is reported for ProCARE because baselines use different interfaces and are not directly comparable at the prompt level. Profiling is a reusable dataset-level cost. For the primary HCC schema, profiling uses approximately $T_{\text{prof}} \approx 65\text{k}$ tokens, and each query uses approximately $T_{\text{inf}} \approx 38\text{k}$ tokens for IR compilation, repair prompts, and certified reporting. For N tasks on the

Table 33: Instantiated report-task DAG for the recurrence-free survival benchmark instance. Each node declares required inputs and produced artifacts. Edges encode artifact prerequisites; infeasible edges are not selected and instead propagate limitations.

Node	Task	Required inputs	Produced artifacts	Report role
T0	Objective binding	User prompt, heterogeneous profile, report schema	Objective, endpoint definition, required section plan	Study overview
T1	Cohort and attrition	`SG.Subject_Id`, `SG.SGDAT`	Analysis cohort, attrition table, index-date assumptions	Overview / Attrition
T2	Baseline covariates	T1, `DM`, `BCLC`, `CP`, `ECOG`	Table 1, missingness table, covariate map	Baseline characteristics
T3	Endpoint and censoring	T1, `VT.PADHDAT`, `VT.PADRDAT`, `VT.VTDAT1`, `VT.VTDAT2`	RFS dataset, event/censor flags, temporal validity log	Primary analysis
T4	Exposure matrix	T1, `TACE`, `TART`, `IMU`, `PATE`	Treatment exposure matrix, dated exposure availability	Primary / Subgroup
T5	Method feasibility gate	T2, T3, T4, procedural policy	KM feasibility decision and caveats	Methods
T6	Primary RFS analysis	T3, T4, T5	KM curve, RFS summary table, primary result text	Primary analysis
T7	Subgroup/sensitivity	T2, T4, T6	Subgroup tables and caveats	Sensitivity analysis
T8	Safety evidence check	`AE_HQ`, `SAE_HQ`, `AE`, `SAE`	Safety limitation statement; blocked detailed AE profiling	Safety / Limitations
T9	Report synthesis	Prior artifacts and evidence map	Final structured report and claim-evidence map	Final report

Table 34: Selected dependency edges and support scores for the instantiated DAG. The feasibility gate s_{ij} blocks hard-infeasible dependencies. The support score c_{ij} summarizes deterministic evidence from variable availability, temporal validity, sample adequacy, and method-design consistency.

Edge	Dependency rationale	$\mathbf{s_{ij}}$	$\mathbf{c_{ij}}$
T0 $\rightarrow$ T1	Cohort construction requires the objective, index date, and output contract.	1	1.00
T1 $\rightarrow$ T2	Baseline summaries must be computed within the index cohort; BCLC and Child–Pugh coverage is high, while age missingness must be disclosed.	1	0.89
T1 $\rightarrow$ T3	RFS endpoints and censoring must be aligned to surgery date; 1,215 subjects have usable follow-up and one temporal anomaly is flagged.	1	0.98
T1 $\rightarrow$ T4	Treatment exposure grouping must be aligned to the index cohort; treatment proxies cover 1,104 subjects but dated exposures are incomplete.	1	0.85
T5 $\rightarrow$ T6	The primary RFS analysis is selected only after the method gate passes; the combined gate has $S_{\mathrm{var}} = 0.828$, $S_{\mathrm{temp}} = 0.999$, $S_{\mathrm{size}} = 1.000$, and $S_{\mathrm{cons}} = 1.000$.	1	0.931
T8 $\rightarrow$ detailed AE profiling	Detailed AE/SAE profiling requires non-empty detail tables; both detail tables contain no valid records.	0	0.00
T8 $\rightarrow$ T9	The empty AE/SAE detail tables support a limitation statement rather than unsupported safety claims.	1	1.00

same dataset,

$$\bar{T}(N) = T_{\mathrm{inf}} + T_{\mathrm{prof}}/N.$$

Thus $\bar{T}(1) \approx 103\mathrm{k}$ tokens, $\bar{T}(20) \approx 41.3\mathrm{k}$ tokens, and $\bar{T}(36) \approx 39.8\mathrm{k}$ tokens per query.

Table 35: Required RWS report sections used by the RS metric.

Section	Expected content
Study Overview	Cohort definition and objective statement.
Attrition Analysis	Patient disposition and eligibility filtering steps.
Baseline Characteristics	Demographics, clinical history, baseline vitals, and baseline labs.
Primary Analysis	Main efficacy or exposure-outcome results, such as KM curves or hazard ratios.
Safety Profiling	Adverse-event screening summaries and available safety proxies.
Sensitivity/Subgroup Analysis	Checks across predefined strata.
Limitations	Data constraints, method limitations, and potential biases.

Table 36: Baseline adaptation and scoring rules.

Method family	Run protocol	Scoring rule
ReAct and ToolLLM	Agentic tool-use baselines are asked to inspect the tables, write executable analysis code, generate artifacts, and produce a report under the same prompt.	Deterministic validators execute the final entry point when provided; reports are checked post hoc for evidence support.
TableGPT-Agent and ChatExcel-Max	Table-analysis agents are asked to generate the requested analysis and report.	If no executable artifact bundle is produced, execution-dependent metrics are N/A; report and claim metrics are computed when possible.
Gemini-3-Pro direct baseline	Gemini-3-Pro receives the task and data-access instructions and is asked to return code, artifacts, and report content.	Generated code is executed in the same sandbox and evaluated by the same artifact and claim validators.
TableGPT2	The table model is evaluated on table-analysis outputs and executable artifacts when available.	Metrics requiring a complete bundle are N/A when the method does not emit the required artifact.
Claude Code and Codex	Native coding-agent baselines receive the benchmark prompt, data-access path, artifact contract, and sandbox budget through their repository editing interfaces. They may create, edit, execute, and repair files within the job workspace.	The final workspace is re-executed from a clean scorer sandbox. Execution, artifact, clinical/statistical language, and claim-trace metrics are computed post hoc from the submitted files and report.
ProCARE	Full profile–IR–execution–certificate–DAG workflow.	All deterministic and judge metrics are computed on the accepted artifact bundle.

D.2 Systems and baseline adaptation

All methods receive the same natural-language task prompt, raw data access boundary, deliverable request, and inference budget. Direct Gemini-3-Pro baselines and direct Gemini-3-Pro scaffolded controls use the same model as the default ProCARE backbone. Claude Code and Codex are evaluated as native code-agent baselines: they can edit files and run code through their coding-agent interfaces, but they do not receive ProCARE's profile object, typed RWS-IR, evidence-certificate admission policy, or certified reporting DAG unless they infer or generate analogous files from the prompt. ProCARE additionally constructs and uses the heterogeneous profile, typed RWS-IR, execution validators, evidence certificates, and certified reporting DAG. Baselines are evaluated under a post-hoc contract checker so that schema use, execution, artifact completeness, and claim traceability are measured against the same profile-derived constraints.

A method is not penalized twice for a missing executable bundle. XE is zero when an executable entry point is submitted but fails, while metrics that require absent artifacts are marked N/A. Schema feasibility, report structure, scientific language, and post-hoc claim diagnostics are still computed whenever enough submitted content is available. Code-agent baselines typically produce executable bundles and stronger local repairs, but their reports are still scored without granting credit for ProCARE-only objects such as accepted RWS-IRs or admitted evidence certificates.

D.3 Fairness-controlled scaffolded baselines

To address the concern that the artifact contract may favor ProCARE, we conducted a scaffolded-baseline evaluation under the same final checker used for the main baseline comparison. These RQ1 fairness controls differ from the RQ2 mechanism ablations in Appendix D.4: ablations remove one component from ProCARE, whereas scaffolded baselines add evaluator-facing information and output schemas to external baselines or a direct Gemini-3-Pro run. Direct Gemini-3-Pro controls use the same provider endpoint, model identifier, decoding settings, and inference budget as the default ProCARE backbone. The scaffold gives non-ProCARE methods the same schema profile

Table 37: Inputs and controls used in the scaffolded-baseline fairness evaluation. These are baseline adaptations, not ProCARE component ablations.

Condition	Information given to model	Output and repair control	Purpose
Profile/template baseline scaffold	Task prompt, raw data access, Φ summary, table/field dictionary, join keys, temporal-role candidates, and artifact template.	One initial generation; no validator feedback loop beyond final scoring.	Controls for missing schema and output-format information.
Profile/template/repair baseline scaffold	Same as above.	Three edit-run-repair turns with standardized validator feedback and identical execution budget.	Controls for unequal repair opportunity.
Direct Gemini-3-Pro + validator interface	Gemini-3-Pro receives Φ, artifact template, sandbox logs, and validator feedback.	Three repair turns; no typed IR compiler or certificate admission gate.	Controls for deterministic validation feedback without ProCARE planning.
Direct Gemini-3-Pro + study-spec schema	Gemini-3-Pro receives Φ and must emit `study_spec.json` before writing code and report.	No ProCARE IR compiler; validator feedback is limited to schema/artifact execution errors.	Controls for evaluator-compatible study-spec output.
Direct Gemini-3-Pro + claim-trace schema	Gemini-3-Pro receives Φ, artifact template, and claim-trace file schema.	No certificate admission gate or certified DAG ordering.	Controls for evaluator-compatible claim-trace output.
ProCARE reference	Full profile, IR compiler, violation-guided execution, evidence certificates, and certified report DAG.	Same three repair turns and final checker.	Reports the integrated contract workflow for comparison.

Table 38: Full metric results for fairness-controlled scaffolded baselines. Higher is better except UCR and CLV. Rows are baseline adaptations, not ProCARE ablations; values are final checker outputs.

Baseline condition	Performance								Contract diagnostics			
	GA↑	PF↑	XE↑	CQ↑	VC↑	CF↑	RS↑	SL↑	VRR↑	CTC↑	UCR↓	CLV↓
ReAct + fairness scaffold	0.82	0.79	0.74	0.69	0.62	0.65	0.75	0.70	0.62	0.70	0.13	0.10
ToolLLM + fairness scaffold	0.79	0.80	0.68	0.65	0.58	0.62	0.72	0.67	0.58	0.66	0.15	0.12
Direct Gemini-3-Pro + profile/template	0.77	0.73	0.66	0.62	0.54	0.58	0.73	0.65	0.31	0.55	0.20	0.15
Direct Gemini-3-Pro + validator repair	0.80	0.78	0.79	0.72	0.60	0.64	0.75	0.69	0.68	0.65	0.15	0.11
Direct Gemini-3-Pro + study-spec schema	0.82	0.82	0.70	0.68	0.57	0.62	0.71	0.68	0.43	0.52	0.21	0.16
Direct Gemini-3-Pro + claim-trace schema	0.78	0.74	0.63	0.61	0.59	0.63	0.74	0.73	0.35	0.74	0.11	0.08
ProCARE reference	**0.86**	**0.84**	**0.92**	**0.88**	**0.72**	**0.74**	**0.80**	**0.79**	**0.81**	**0.90**	**0.06**	**0.03**

Φ, artifact-bundle template, raw data access boundary, execution sandbox, and three-turn repair budget needed to satisfy the evaluator. It does not give baselines ProCARE's IR compiler, certificate admission gate, certified reporting DAG, or hidden planner state. Rows that mention a study-spec or claim-trace schema only require compatible JSON files for scoring.

The shared artifact template requires: `study_spec.json`, `analysis.py` or `main.py`, `result_manifest.json`, `figure_manifest.json`, `claim_trace.json` when the method attempts explicit claim tracing, `assumptions.json`, and `report.md`. For methods that do not emit a typed RWS-IR, the evaluator maps the submitted `study_spec.json` into the canonical scoring schema before computing PF. For methods that do not emit explicit evidence certificates, CTC/UCR/-CLV are computed by the same post-hoc evidence reconstruction used in Table 36.

Table 38 reports the final checker outputs for the scaffolded-baseline evaluation. Scaffolded baselines improve substantially over generic agents, reducing N/A rates and narrowing the execution gap, while the integrated ProCARE workflow remains strongest on violation resolution and claim-level traceability. The table should be read as a fairness-control comparison, not as evidence that individual ProCARE modules have been added to an otherwise unchanged LLM. All direct rows use Gemini-3-Pro.

Table 39: Ablation definitions for contract mechanisms.

Variant	Removed component	What remains fixed
w/o profile Φ	The model does not receive the heterogeneous profile as a contract object. It may inspect raw table metadata and generated summaries, but generation lacks profile-grounded repair hints.	Prompt, budget, execution sandbox, reporting request, and post-hoc evaluator.
w/o RWS-IR	The model writes code and a report without producing the typed RWS-IR.	Raw data access, profile summaries when available, execution sandbox, and post-hoc claim diagnostics. PF is N/A because no typed IR exists.
w/o violation-guided repair	Runtime validators detect violations, but localized violation sets are not fed back into the next edit.	Profile, RWS-IR, execution sandbox, artifact validators, certificates, and report policy.
w/o evidence certificates	The report is generated without explicit claim certificates and without the claim-admission gate.	Profile, RWS-IR, execution repair, artifacts, and DAG section ordering. CTC for emitted explicit certificates is zero; post-hoc support can still be inspected separately.
w/o certified DAG ordering	Evidence certificates remain available, but report sections are generated in a fixed or unconstrained order without dependency-aware traversal.	Profile, RWS-IR, execution repair, artifact validators, and claim admission.
Full ProCARE	No contract component is removed.	Complete profile–IR–repair–certificate–DAG workflow.

D.4 Ablation protocol

Mechanism ablations remove one contract component at a time while keeping all remaining components fixed whenever the remaining workflow is defined. Table 39 specifies the intervention behind the ablation table in the main paper.

D.5 Model access, decoding, and version identifiers

Backbone-sensitivity experiments keep profiling, RWS-IR compilation, violation-guided execution, certified reporting, prompts, and scoring scripts fixed. The table records the paper-level model identifiers and roles used in the experiments. Provider-returned request metadata is retained in run logs; when a provider alias differs from the paper-level display name, the logged identifier is the reproducibility target. Generation runs use the released implementation's production decoding settings, with temperature 0.2 for Gemini/openai-compatible providers and 0.1 for DeepSeek-compatible providers; judge models are run at temperature 0.

D.6 Reproducibility package

The repository release contains the manuscript sources, system code, prompt definitions, the fully de-identified HCC benchmark tables, Public-RWS bundle metadata, profile artifacts, evaluation scripts, and figure-generation scripts. The HCC release includes the record-level tables needed to rerun the 36 primary tasks, with direct identifiers removed and random benchmark IDs retained for reproducible linkage.

Listing 6: Local reproduction commands.

```
cd code/backend
python -m app.cli --excel <workbook.xlsx> --topic "<benchmark prompt>" --refine
python tests/run_all.py

cd ../../paper
build.cmd nips appendix
```

E Additional Results and Diagnostics

E.1 Claim-diagnostic interpretation

More specific prompts increase claim traceability because they constrain variables, temporal anchors, methods, and expected artifacts before reporting. The dominant downgrades are language-policy failures rather than arithmetic inconsistencies: causal phrasing from descriptive analyses, safety

Table 40: Model access, version metadata, and evaluation roles.

Role	Paper-level model identifier	Access mode	Use in evaluation
Backbone	GPT-5.2	Proprietary API	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone	GPT-4 Turbo	Proprietary API	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone	Claude Opus 4.5	Proprietary API	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone	Claude Sonnet 4.5	Proprietary API	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone	Claude-3 Opus	Proprietary API	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone, direct baseline, and scaffolded control	Gemini-3-Pro	Proprietary API	Default ProCARE backbone; also used for the direct baseline and all Direct Gemini-3-Pro scaffolded fairness-control rows.
Backbone and judge	Gemini-2.5-Pro	Proprietary API	Backbone-sensitivity run and one blinded semantic judge.
Backbone	Qwen3-Max	API endpoint	Backbone-sensitivity run under the fixed ProCARE workflow.
Backbone	`Qwen3-235B-A22B-Instruct-2507`	API endpoint or hosted open-source model	Backbone-sensitivity run.
Backbone	DeepSeek-V3.2	API endpoint or hosted open-source model	Backbone-sensitivity run.
Backbone	DeepSeek-R1-0528	API endpoint or hosted open-source model	Backbone-sensitivity run and non-valid-bundle baseline category.
Code-agent baseline	Claude Code (Claude Sonnet 4.5 backend)	Proprietary coding-agent interface	Baseline run through the native Claude Code repository-editing workflow; provider-returned CLI/model metadata is retained in the run log.
Code-agent baseline	Codex (`gpt-5.3-codex` backend)	Proprietary coding-agent interface	Baseline run through the native Codex repository-editing workflow; CLI session metadata and final workspace state are retained for reproduction.
Judge	Qwen2.5-1M	API endpoint	Second blinded semantic judge.

Table 41: Repository components used for reproduction.

Path	Contents
`paper/`	Manuscript, supplement, bibliography, NeurIPS style file, build script, and paper figures.
`code/backend/`	Profiling, planning, code-generation, execution, report-generation, and API services.
`code/backend/tests/`	Local tests for profile construction, planning, code execution, OpenCode integration, and report generation.
`code/frontend/`	Interface for project setup, profile inspection, RWS-IR review, execution monitoring, and report review.
`code/output/knowledge_base/profiles/`	Cached profile artifacts used by the implementation.
`rws_hcc_benchmark/`	Fully de-identified HCC benchmark tables, data dictionary, checksums, task metadata, and release terms for reproducing the 36 primary tasks.
`rws_public_datasets_bundle/`	Public-RWS source manifests, checksums, table shapes, and public dataset files.
`figure_code/`	Scripts used to regenerate paper diagnostic figures.

claims based only on screening pages, or subgroup conclusions without sufficient support. During certification, these claims are rewritten as limitations or removed from the admitted factual-claim set.

E.2 Backbone-sensitivity interpretation

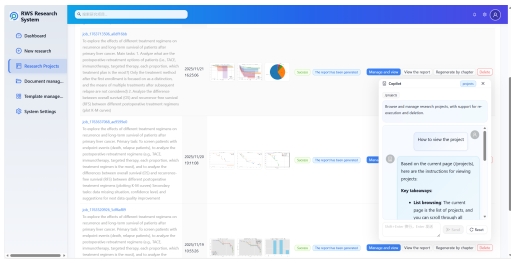
Backbone substitution changes only the base LLM used for planning, code editing, and report drafting. Profiling, RWS-IR schema, validators, evidence-certificate admission, report schema, and evaluation remain fixed. Stronger backbones improve GA, PF, VRR, and CTC because they use profile constraints and repair feedback more reliably. Execution-only differences are smaller than planning and reporting differences because deterministic validation and localized repair bound the accepted artifact space after code is generated.

F System Interface and Workflow

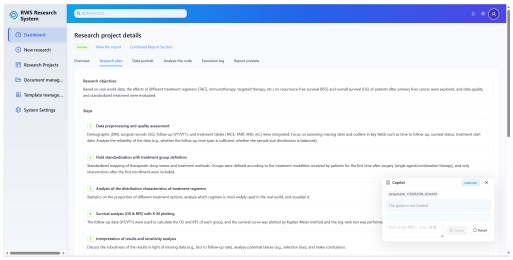
Figures 4 and 5 show the implementation interface states. They are included as workflow evidence rather than method exposition: the panels identify where profile artifacts, IR traces, execution logs, generated figures, and section-level report blocks are inspected during a run.



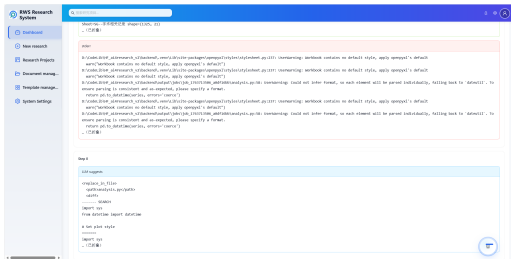
(a) System dashboard for managing RWS projects and cohort status.



(b) Profile panel for reviewing structural metadata (Φ^S) and ontology mappings (Φ^K).



(c) RWS-IR view showing goal compilation ($T_{\text{goal}} \to Z$).

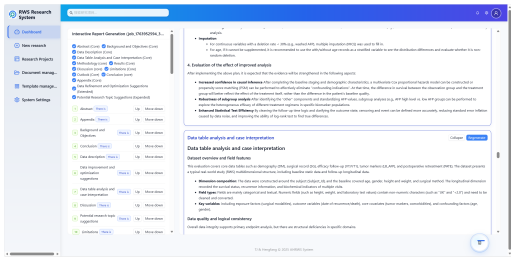


(d) Coding harness view with reasoning traces, sandbox logs, and verification status.

Figure 4: ProCARE interface states for project setup, profile inspection, RWS-IR compilation, and coding execution.



(a) Report review view with generated tables, charts, and evidence links.



(b) Section-wise report generation with data-bound report blocks.

Figure 5: Report-facing interface states for reviewing generated artifacts, evidence links, and section-wise report content.

G Evaluation Protocol and Rubrics

Evaluation uses deterministic validators for feasibility and execution checks, and LLM judges for semantic quality.

G.1 Metrics Overview

Table 42 defines the metrics used for end-to-end RWS automation. All scores are normalized to $[0, 1]$.

G.2 Deterministic validators

Deterministic validators are used whenever the property can be checked from the artifact bundle. PF checks whether the accepted RWS-IR references only profiled tables, fields, keys, clinical roles, temporal anchors, and methods. XE checks whether the submitted entry point exits successfully, avoids failure sentinels, and writes the required result and figure manifests. Artifact completeness requires declared output files to exist, be nonempty, and be referenced by the corresponding manifest.

Table 42: Metrics used in the appendix evaluation protocol.

Metric	Type	Scoring rule
RWS-IR Feasibility (PF)	Deterministic + judge	Validates the RWS-IR against Φ^S and penalizes non-existent tables, columns, invalid joins, unsupported temporal anchors, or incompatible analysis grains. Residual ambiguity is checked by the judge.
Code Executability (XE)	Deterministic	Sandbox execution success rate, n_{pass}/N; runtime errors and invalid output files count as failures.
Violation Resolution Rate (VRR)	Deterministic	Fraction of detected execution-time contract violations resolved before the accepted run.
Report Structure (RS)	Deterministic	Fraction of required report sections in Table 35 covered with non-trivial content.
Claim Trace Coverage (CTC)	Deterministic	Fraction of final factual claims linked to valid evidence certificates.
Unsupported Claim Rate (UCR)	Deterministic	Fraction of final factual claims without admissible evidence support.
Claim Language Violation (CLV)	Deterministic + judge	Fraction of factual claims whose wording exceeds the support level allowed by the evidence certificate and RWS-IR claim policy.
Goal Alignment (GA)	Judge	Alignment with the clinical objective, target population, and requested endpoint.
Code Quality (CQ)	Judge	Soundness of data handling, missingness handling, joins, and statistical implementation.
Visual Clarity (VC)	Judge	Readability and labeling quality of generated figures.
Chart Fidelity (CF)	Judge	Consistency between plotted values, execution logs, and report statements.
Scientific Language (SL)	Judge	Clinical tone, hedging, and limitation disclosure.

Temporal consistency checks row-level ordering between index, exposure, event, and censoring dates when those roles are declared in Z. Chart-result consistency checks whether plotted variables, group labels, sample sizes, and reported summary values match execution logs or result tables.

All metrics are normalized to $[0, 1]$. XE is n_{pass}/N over tasks. PF is one for a statically admissible RWS-IR, zero for a hard-invalid RWS-IR, and partially credited only for residual semantic ambiguity adjudicated by the fixed rubric. RS is the fraction of required report sections with task-relevant content. VRR, CTC, UCR, and CLV are computed from logged violations and extracted factual claims as described in Appendix C.5 and Appendix C.6.

For baselines that do not implement ProCARE's contract internally, the evaluator reconstructs the same diagnostics from final artifacts. A baseline can receive CTC credit only when a factual claim can be linked to an executed result, code region, profile-supported field, or explicit assumption. Missing artifacts or unsupported variables block traceability.

G.3 LLM judge protocol

We use *Gemini-2.5-Pro* and *Qwen2.5-1M* as independent judges with temperature 0. Judges are blinded to method identity and receive the same artifact bundle: input prompt, generated RWS-IR, code, execution logs, result and figure manifests, evidence certificates when available, assumptions, limitations, and final report. Scores are normalized to $[0, 1]$ and averaged across judges. Schema hallucination and artifact inconsistency are handled as hard rubric failures for the affected dimensions.

G.4 Expert-review protocol

Expert review is performed on the 36 primary HCC benchmark instances. The panel contains five blinded reviewers: three board-certified clinical experts with hepatobiliary oncology or hepatology experience (C1–C3) and two statistics/data experts with clinical data-analysis experience (S1–S2). Reviewers receive anonymized artifact bundles containing the task prompt, study specification,

Table 43: Raw expert-rating data format used before aggregation. One row stores one reviewer–instance–stage score.

Column	Allowed values or format	Definition
review_id	stable string	Unique row identifier.
benchmark_id	HCC-001–HCC-036	Primary benchmark instance identifier.
objective	cohort, safety-proxy, efficacy, resource-use	Benchmark objective family.
prompt_level	goal, goal+approach, goal+method	Prompt-specificity level.
method_id	submitted method identifier	Method label masked during review and restored only for analysis.
stage	rws_ir, code, visualization, reporting	Workflow stage being scored.
artifact_id	path or hash	Submitted artifact bundle, code file, figure set, or report identifier.
reviewer_id	C1, C2, C3, S1, S2	De-identified reviewer identifier.
reviewer_role	clinical or statistics_data	Reviewer expertise group.
criterion_id	global_stage_score or rubric subcriterion	Overall stage score or optional subcriterion.
score_1_5	integer 1–5	Raw Likert score before normalization.
major_error_flag	0 or 1	Reviewer-marked hard error, such as invalid endpoint, impossible join, or irreproducible code.
claim_flag	0, 1, or NA	Whether the reviewed report or figure interpretation contains a claim exceeding evidence support.
comment_code	none, schema, method, temporal, claim, figure, reproducibility, other	Structured reason for downgrade, if any.
free_text_comment	de-identified text	Short reviewer note used for adjudication and error analysis.
timestamp_utc	ISO-8601 timestamp	Completion time of the review row.

code or execution logs when relevant, figures, assumptions, limitations, evidence certificates when available, and final report. Method identity is masked in the review package.

Reviewers score each stage on a five-point Likert scale: 1 indicates an invalid or irrelevant artifact, 3 indicates a partially correct artifact with important limitations, and 5 indicates a clinically and methodologically appropriate artifact with limitations disclosed. The clinical panel focuses on clinical objective alignment, endpoint and covariate validity, clinical interpretability of figures, and claim scope. The statistics/data panel focuses on data linkage, analysis grain, statistical method choice, code reproducibility, figure-result consistency, and whether reported results are supported by executed artifacts. Code scores in the main table use the statistics/data panel only; the other stage scores aggregate all five reviewers because both clinical and statistical/data judgments are relevant.

The raw evaluation data are represented in the long format in Table 43. The required stage-level row set contains $36 \times (5 + 2 + 5 + 5) = 612$ reviewer–instance–stage records before optional criterion-level comments: five reviewers score RWS-IR, visualization, and reporting, while S1–S2 score code implementation. Additional criterion-specific rows can be added with the same keys when a reviewer records a subscore or a major error flag.

For each benchmark instance i, reviewer r, and stage s, the raw Likert score is $x_{irs} \in \{1, \ldots, 5\}$. Stage-level expert scores are first averaged across the eligible reviewer set R_s:

$$E_{is} = \frac{1}{|R_s|} \sum_{r \in R_s} x_{irs}.$$

Scores are normalized to $[0, 1]$ for calibration plots by $\tilde{E}_{is} = (E_{is} - 1)/4$, but the main paper reports mean$\pm$SD on the original 1–5 scale. Spearman ρ is computed between per-instance expert scores and the corresponding automated stage score: $(\mathrm{GA} + \mathrm{PF})/2$ for RWS-IR, CQ for code, $(\mathrm{VC} + \mathrm{CF})/2$ for visualization, and SL for reporting. Confidence intervals for mean expert scores use a t interval over 36 instance-level means; confidence intervals for Spearman correlations use Fisher transformation.

G.5 Semantic Rubric Anchors

Judges assign 1–5 Likert scores for Goal Alignment (GA), Code Quality (CQ), Visual Clarity (VC), and Scientific Language (SL). Scores are normalized to $[0, 1]$ with a linear transformation. Table 48 lists the anchor definitions.

Table 44: Expert panel responsibilities and raw scoring volume.

Stage	Eligible reviewers	Primary review focus	Raw records
RWS-IR	C1–C3, S1–S2	Clinical goal, endpoint validity, data linkage, analysis grain, and statistical design feasibility.	180
Code	S1–S2	Reproducibility, executable implementation, statistical method use, artifact writing, and result consistency.	72
Visualization	C1–C3, S1–S2	Clinical interpretability, sample-size labeling, figure-result agreement, and avoidance of unsupported visual claims.	180
Reporting	C1–C3, S1–S2	Clinical language, limitation disclosure, statistical wording, and claim-to-evidence support.	180

Table 45: Stage-level expert score summary on the 36 primary HCC benchmark instances. CI is the 95% confidence interval for the instance-level mean.

Stage	Instance means	Raw records	Mean±SD	95% CI	Median [IQR]
RWS-IR	36	180	4.38±0.48	4.22–4.54	4.33 [4.00, 4.67]
Code	36	72	4.42±0.21	4.35–4.49	4.42 [4.25, 4.58]
Visualization	36	180	4.50±0.53	4.32–4.68	4.67 [4.00, 5.00]
Reporting	36	180	4.13±0.64	3.91–4.35	4.17 [3.67, 4.67]

Table 46: Inter-rater reliability for the human expert panel. Stages scored by five reviewers use two-way random-effects ICC for averaged ratings; the two-reviewer code stage uses weighted Cohen's κ_w.

Stage	Panel	Statistic	Estimate	95% CI	Interpretation
RWS-IR	C1–C3, S1–S2	ICC(2,k)	0.84	0.72–0.91	Strong agreement on specification validity.
Code	S1–S2	κ_w	0.76	0.61–0.88	Substantial agreement on reproducibility and statistical implementation.
Visualization	C1–C3, S1–S2	ICC(2,k)	0.79	0.66–0.88	Strong agreement on figure clarity and consistency.
Reporting	C1–C3, S1–S2	ICC(2,k)	0.77	0.63–0.87	Substantial-to-strong agreement on language and claim scope.

Table 47: Correlation between automated metrics and aggregated expert scores. q is the Benjamini–Hochberg adjusted p value across the four stage-wise correlations.

Stage	Expert aggregation	Automated score	ρ	95% CI	p	q
RWS-IR	Mean of C1–C3, S1–S2	$(GA + PF)/2$	0.82	0.67–0.90	< 0.001	< 0.001
Code	Mean of S1–S2	CQ	0.74	0.54–0.86	< 0.001	< 0.001
Visualization	Mean of C1–C3, S1–S2	$(VC + CF)/2$	0.75	0.55–0.86	< 0.001	< 0.001
Reporting	Mean of C1–C3, S1–S2	SL	0.71	0.49–0.84	< 0.001	< 0.001

H Prompts and Templates

H.1 Benchmark prompt template

Each benchmark instance is rendered into a common prompt with three fields: **Task**, optional **Guidance**, and **Deliverables**. The task states the study objective. Guidance supplies approach-level or method-level constraints for GOAL+APPROACH and GOAL+METHOD. Deliverables require a validated RWS-IR, one executable entry point with generated artifacts, and a structured report with assumptions, limitations, and evidence certificates. Hallucinated tables or columns are disallowed.

H.2 IR compiler prompt template

The IR compiler receives the user prompt, high-level structural profile, high-level knowledge profile, output contract, and claim-policy rules. It returns JSON containing study design, analysis grain, join graph, entity keys, temporal anchors, variable-role bindings, method specification, required artifacts, assumptions, and allowed or blocked claim types. If the profile does not support a requested variable or method, the compiler must either downgrade the task or return an infeasibility trace.

Table 48: Semantic rubric anchors for LLM judges.

Metric	Score 5	Score 3	Score 1
Goal Alignment	Addresses the clinical objective, target population, and endpoints with explicit limitation disclosure.	Partially addresses the task with minor errors in inclusion criteria or endpoints.	Gives an irrelevant analysis or misunderstands the clinical goal.
Code Quality	Uses sound pandas or SQL logic, handles missing data and types correctly, and keeps the implementation modular.	Runs but remains fragile, often because of hardcoded values or inefficient joins.	Uses an incorrect method, applies the wrong statistical test, or contradicts the plan.
Visual Clarity	Produces figures with clear axes, legends, labels, and titles that support the reported results.	Produces legible charts with missing units, overlapping labels, or default styling.	Produces misleading, illegible, or unrelated visualizations.
Scientific Language	Uses clinical language with appropriate hedging and clear limitation disclosure.	Uses a general scientific tone but has verbose phrasing or imprecise medical terminology.	Uses casual language or makes unsupported claims.

H.3 Coding harness prompt template

The coding harness receives the accepted RWS-IR, low-level field profile, selected sheets, local figure policy, artifact schema, and any previous violation set. It writes one executable entry point, generates declared artifacts, and produces figure and result manifests. Repairs must address the localized violation set without introducing fields or methods absent from the profile.

H.4 Report and certificate prompt template

The report generator receives the accepted RWS-IR, result artifacts, figure manifest, execution logs, assumptions, limitations, and claim-policy rules. Each factual sentence must be paired with an evidence certificate or rewritten as a limitation. The generator must not convert descriptive proxy artifacts into causal, safety-critical, regulatory, or treatment-effect claims.

H.5 Judge prompts

The judge receives the ground-truth data profile Φ so that schema hallucinations can be checked against the same table and field constraints used by the validators.

System prompt. Defines the reviewer role and scoring constraints.

Prompt

Prompt 1. Expert Judge System Prompt
You are a Senior Clinical Data Scientist reviewing Real-World Study (RWS) automation artifacts.
Your task is to evaluate an AI agent's artifacts against a specific **Ground Truth Schema**.
Scoring rules:
1. **Schema Hallucination = 0:** Any usage of tables/columns not present in the provided `[DATA_PROFILE]` results in a score of 0 for relevant metrics, regardless of code quality.
2. **Consistency:** The numbers in the Final Report must be consistent with the Execution Logs.
3. **Structured rationale:** Provide a concise evidence-based rationale before assigning JSON scores.

User prompt. Injects the profile and artifact bundle.

Prompt

Prompt 2. Context-Injection User Prompt
[PART 1: GROUND TRUTH CONTEXT]

USER QUERY: {user_query}

DATA PROFILE CONSTRAINTS (Φ):
(Valid tables, columns, and join keys derived from the dataset)
{profile_snapshot_str}

[PART 2: AGENT ARTIFACTS]
1. RWS-IR: {agent_rws_ir_json}
2. CODE: {generated_code}
3. LOGS: {execution_logs}
4. VISUALS: {list_of_generated_files}
5. EVIDENCE CERTIFICATES: {evidence_certificate_json}
6. REPORT: {final_report_content}

[PART 3: TASK]
Analyze the artifacts and score them based on the rubric.
Structured assessment
- *Schema Check:* Does the code use ONLY columns listed in [Data Profile Constraints]?
- *Logic Check:* Is the statistical method appropriate for {user_query}?
- *Consistency:* Do chart findings match the log data?

JSON output
Return a JSON object with keys: GA, CQ, VC, CF, SL.
Format: `{"Analysis": "...", "Scores": {...}, "Failure_Mode": "None/Hallucination/Logic"}`

I Data Access, Privacy, and Responsible Use

The released materials include the reproducible benchmark logic and the fully de-identified record-level HCC benchmark tables. The public package includes prompts, profiles, schema artifacts, aggregate summaries, de-identified HCC tables, HCC data dictionary, checksums, public-data manifests, and evaluation scripts. Direct participant identifiers and identity-linking fields are removed from the HCC release, while random benchmark subject and visit identifiers are retained to preserve reproducible table joins and longitudinal task execution. Public-RWS datasets are accessed from their original public sources, with source metadata and checksums recorded in the release documentation.

The HCC benchmark is released for research reproduction and auditing of RWS automation methods. The release terms prohibit attempted re-identification, participant linkage to external data sources, and clinical use of generated outputs. Any downstream deployment on institutional data requires local governance review, schema revalidation, and confirmation that the site-specific claim policy matches the available data.

ProCARE is evaluated as a research automation and auditability system, not as an autonomous clinical decision system. Evidence certificates establish traceability to implemented validators and artifacts. They do not establish clinical truth, causal validity, regulatory sufficiency, or prospective safety. The primary HCC benchmark is single-center and disease-specific. Public-RWS-18 tests schema transfer, but it does not replace multi-institutional longitudinal RWS validation. Porting ProCARE to OMOP, FHIR, or another institutional schema requires rebuilding the profile, revalidating RWS-IR checks, and confirming that claim policies match the new data source and study design.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the profile-grounded contract, typed RWS-IR, profile-constrained execution, evidence certificates, and benchmark scope; the experiments and appendix support these claims with the 36-instance HCC benchmark, Public-RWS-18 validation, fairness-controlled scaffolded baselines, mechanism ablations, five-expert review, and statistical analyses.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The main paper includes a limitations paragraph covering the single-center HCC benchmark, Public-RWS transfer boundary, automated scoring sensitivity, validator coverage, OMOP/FHIR portability constraints, and future multi-institution validation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper does not claim theoretical guarantees or present theorems; the formulas specify procedural artifacts, validation gates, and evaluation metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper and supplementary appendix describe the dataset schema, the 36 HCC benchmark instances, Public-RWS-18 tasks, output contracts, scaffolded-baseline fairness protocol, artifact templates, evaluation metrics, judge prompts, expert raw-rating format, model metadata, and execution constraints; code, benchmark assets, and evaluation scripts are linked through the repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The abstract points to an public repository containing code, the fully de-identified HCC benchmark tables, benchmark prompts, public-data source manifests, profile artifacts, artifact-bundle templates, and evaluation scripts; the appendix specifies benchmark construction, de-identification, release contents, and reproduction commands.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experiments specify benchmark tasks, model backbones, prompt specificity levels, baseline adaptation rules, scaffolded fairness controls, artifact templates, three-turn repair budgets, validator settings, judge settings, inference budget, and evaluation metrics. There is no model training in the proposed system.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: The paper and appendix report mean±SD and 95% confidence intervals for expert scores, inter-rater reliability with 95% confidence intervals, Spearman correlations with 95% confidence intervals, p values, and Benjamini–Hochberg adjusted q values for automated-vs-expert agreement. Deterministic benchmark metrics are averaged over the fixed task suites with full metric definitions and scaffolded-baseline tables.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports token costs for profiling, per-query inference, and reusable profile amortization, and the appendix describes the local Python execution sandbox, validator runs, model-access metadata, and CPU/API-backed execution environment; the experiments rely on API-backed LLMs rather than local model training.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes].

   Justification: The benchmark is derived from IRB-approved HCC data and released only as fully de-identified record-level tables with direct participant identifiers and identity-linking fields removed; it is used for evaluation rather than direct clinical deployment.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The introduction motivates reliable automation for real-world evidence generation, while the method, evaluation, and limitations sections discuss validation constraints, hallucination control, claim traceability, de-identified data-release safeguards, and the need for expert oversight in high-stakes settings.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes].

    Justification: The release uses fully de-identified HCC benchmark tables, aggregate/profile examples, public-data source manifests, checksums, and responsible-use terms. Direct identifiers are removed, random benchmark IDs are retained only for reproducible linkage, and the system includes schema, artifact, certificate, and reporting gates that reject unsupported clinical claims.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper cites the external LLM-agent baselines, public health datasets, clinical-data standards, and model providers used for comparison, validation, or motivation. Public code, de-identified HCC benchmark assets, source manifests, data-use terms, and license information are distributed through the accompanying repository.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The paper introduces the RWS benchmark, fully de-identified HCC benchmark tables, Public-RWS-18 validation suite, system code, artifact-bundle templates, profile artifacts, raw expert-rating data format, and evaluation scripts; the appendix documents task definitions, dataset fields, de-identification and release contents, rubrics, prompts, output contracts, and statistical analyses.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes].

Justification: The work does not use crowdsourcing. For the blinded expert technical validation, the appendix provides the review package contents, reviewer roles, stage responsibilities, raw-rating schema, aggregation rules, reliability analyses, and automated-vs-expert statistical tests; no crowdworker task or crowdworker compensation scheme is involved.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [Yes].

Justification: The benchmark section states that the HCC observational data were collected under institutional review board approval and released only as a fully de-identified benchmark with direct identifiers removed. The expert review uses anonymized artifact bundles and does not expose patient identifiers.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes].

Justification: LLMs are central to ProCARE, the baseline agents, scaffolded fairness controls, and automated judging protocol; the paper reports the evaluated model families, judge models, prompt-specificity settings, fixed workflow components, and validator-constrained repair budget.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.