Evaluation Harnesses and Protocol Design

Deep Dive into DeepSeek — Appendix A

Yin Yang

Foundation Books Project

The one rule of this chapter

A benchmark score is a claim about a piece of software, not just a number.

- Item, prompt, invocation, extraction, scoring, sampling, and aggregation are seven separate stages, and each one can move the final value.

The one rule of this chapter

A benchmark score is a claim about a piece of software, not just a number.

- Item, prompt, invocation, extraction, scoring, sampling, and aggregation are seven separate stages, and each one can move the final value.
- Cost, latency, leakage, prompt sensitivity, and judge calibration are properties of the *protocol*, not decorations added after the score is computed.

The one rule of this chapter

A benchmark score is a claim about a piece of software, not just a number.

- Item, prompt, invocation, extraction, scoring, sampling, and aggregation are seven separate stages, and each one can move the final value.
- Cost, latency, leakage, prompt sensitivity, and judge calibration are properties of the *protocol*, not decorations added after the score is computed.
- A published number that omits any of these stages has been *described*, not *reproduced*.

The one rule of this chapter

A benchmark score is a claim about a piece of software, not just a number.

- Item, prompt, invocation, extraction, scoring, sampling, and aggregation are seven separate stages, and each one can move the final value.
- Cost, latency, leakage, prompt sensitivity, and judge calibration are properties of the *protocol*, not decorations added after the score is computed.
- A published number that omits any of these stages has been *described*, not *reproduced*.

Goal: follow the DeepSeek-R1 AIME 2024 row — 79.8 pass@1, 64 samples per item — from a reported release claim to an executable harness, and see exactly where the executable version can and cannot close the gap.

Roadmap

The protocol is executable software

Benchmarks, prompts, and scoring

Sampling, aggregation, and the unit of independence

Cost, latency, and what a number is allowed to claim

Leakage, aging, and robustness

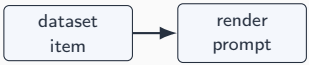
Judges, harness design, and the worked AIME harness

The protocol is executable software

One item, five transformations, one row

dataset
item

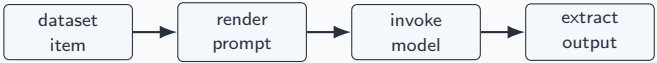
One item, five transformations, one row



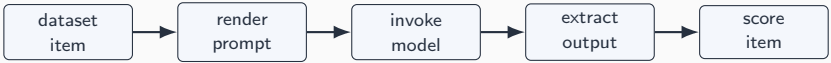
One item, five transformations, one row



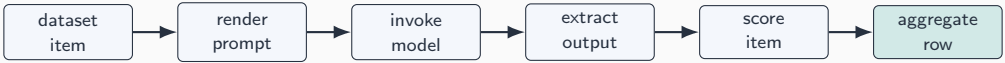
One item, five transformations, one row



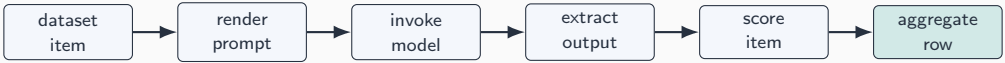
One item, five transformations, one row



One item, five transformations, one row

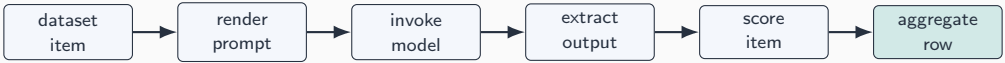


One item, five transformations, one row



- Chapter 2 defines the benchmark protocol and the canonical measurement record. This appendix makes that protocol *executable*.
- Even a result printed as one number in a report is the output of this five-stage pipeline, run once per sampled response.

One item, five transformations, one row



- Chapter 2 defines the benchmark protocol and the canonical measurement record. This appendix makes that protocol *executable*.
- Even a result printed as one number in a report is the output of this five-stage pipeline, run once per sampled response.

Five items with four samples each is twenty invocation, extraction, scoring, and usage records — even when only one answer per item is ultimately displayed.

The running case: DeepSeek-R1 on AIME 2024

Reported quantity	Value
Metric	pass@1
Score	79.8
Maximum generation length	32,768 tokens

The running case: DeepSeek-R1 on AIME 2024

Reported quantity	Value
Metric	pass@1
Score	79.8
Maximum generation length	32,768 tokens
Temperature	0.6
Top-p	0.95
Responses per query	64

The running case: DeepSeek-R1 on AIME 2024

Reported quantity	Value
Metric	pass@1
Score	79.8
Maximum generation length	32,768 tokens
Temperature	0.6
Top-p	0.95
Responses per query	64

[REPORTED: DeepSeek-R1 README / technical report]

The running case: DeepSeek-R1 on AIME 2024

Reported quantity	Value
Metric	pass@1
Score	79.8
Maximum generation length	32,768 tokens
Temperature	0.6
Top-p	0.95
Responses per query	64

[REPORTED: DeepSeek-R1 README / technical report]

- These settings fix parts of the invocation record, but not the prompt template, the item list, the extractor, or the invalid-output policy.

The running case: DeepSeek-R1 on AIME 2024

Reported quantity	Value
Metric	pass@1
Score	79.8
Maximum generation length	32,768 tokens
Temperature	0.6
Top-p	0.95
Responses per query	64

[REPORTED: DeepSeek-R1 README / technical report]

- These settings fix parts of the invocation record, but not the prompt template, the item list, the extractor, or the invalid-output policy.
- A harness must still supply all of those before it can execute "the same protocol" locally.

Benchmarks, prompts, and scoring

A benchmark item is a triple

Definition (Benchmark item)
A benchmark item is $x = (u, y, \eta)$: input u, target or reference artifact y, and metadata η — source, difficulty, date, license, contamination status, task family.

- Math wants a final number; code wants a patch that passes tests; tool-use wants a call trace; preference wants a judged comparison.

A benchmark item is a triple

Definition (Benchmark item)
A benchmark item is $x = (u, y, \eta)$: input u, target or reference artifact y, and metadata η — source, difficulty, date, license, contamination status, task family.

- Math wants a final number; code wants a patch that passes tests; tool-use wants a call trace; preference wants a judged comparison.
- Two rows with the same metric label are different measurement claims if they differ in task family, split, output object, or scorer.

A benchmark item is a triple

Definition (Benchmark item)
A benchmark item is $x = (u, y, \eta)$: input u, target or reference artifact y, and metadata η — source, difficulty, date, license, contamination status, task family.

- Math wants a final number; code wants a patch that passes tests; tool-use wants a call trace; preference wants a judged comparison.
- Two rows with the same metric label are different measurement claims if they differ in task family, split, output object, or scorer.

The dataset and split set the denominator for every later metric — reporting the split is part of the claim, not paperwork around it.

Same label, different rule: extracting 17 vs. 32

Let a math item ask for the integer n with $n \equiv 2 \pmod 5$, $n \equiv 3 \pmod 7$, $0 \leq n < 35$. The answer is 17.

- An output containing only $\boxed{17}$ extracts to 17 and scores 1.

Same label, different rule: extracting 17 vs. 32

Let a math item ask for the integer n with $n \equiv 2 \pmod 5$, $n \equiv 3 \pmod 7$, $0 \leq n < 35$. The answer is 17.

- An output containing only $\boxed{17}$ extracts to 17 and scores 1.
- An output containing only $\boxed{32}$ extracts to 32 and scores 0.

Same label, different rule: extracting 17 vs. 32

Let a math item ask for the integer n with $n \equiv 2 \pmod 5$, $n \equiv 3 \pmod 7$, $0 \leq n < 35$. The answer is 17.

- An output containing only $\boxed{17}$ extracts to 17 and scores 1.
- An output containing only $\boxed{32}$ extracts to 32 and scores 0.
- An output containing *both* values: a strict extractor returns an invalid marker rather than choosing after the fact.

Same label, different rule: extracting 17 vs. 32

Let a math item ask for the integer n with $n \equiv 2 \pmod 5$, $n \equiv 3 \pmod 7$, $0 \leq n < 35$. The answer is 17.

- An output containing only $\boxed{17}$ extracts to 17 and scores 1.
- An output containing only $\boxed{32}$ extracts to 32 and scores 0.
- An output containing *both* values: a strict extractor returns an invalid marker rather than choosing after the fact.

That invalid policy changes the denominator, and it must be fixed *before* the run — not discovered while reading transcripts.

Sampling, aggregation, and the unit of independence

Generations are not new benchmark items

Definition (Unit of independence)
The level at which a new observation contributes new information about the population named by the claim.

- 30 AIME-style items $\times$ 64 generations each is $30 \cdot 64 = 1{,}920$ scored outputs — but still 30 items.

Generations are not new benchmark items

Definition (Unit of independence)
The level at which a new observation contributes new information about the population named by the claim.

- 30 AIME-style items $\times$ 64 generations each is $30 \cdot 64 = 1{,}920$ scored outputs — but still 30 items.
- $\widehat{p}_i = \frac{1}{64} \sum_j r_{i,j}$, then $\widehat{p} = \frac{1}{30} \sum_i \widehat{p}_i$: resample the 30 item records, keeping each item's 64 outputs together.

Generations are not new benchmark items

Definition (Unit of independence)
The level at which a new observation contributes new information about the population named by the claim.

- 30 AIME-style items $\times$ 64 generations each is $30 \cdot 64 = 1{,}920$ scored outputs — but still 30 items.
- $\widehat{p}_i = \frac{1}{64} \sum_j r_{i,j}$, then $\widehat{p} = \frac{1}{30} \sum_i \widehat{p}_i$: resample the 30 item records, keeping each item's 64 outputs together.

Treating all 1,920 labels as independent items would hide the shared difficulty inside each problem and misstate generalization evidence.

Majority vote, best-of-n, self-consistency: not the same metric

- **Majority vote** scores the most frequent extracted answer.

Majority vote, best-of-n, self-consistency: not the same metric

- **Majority vote** scores the most frequent extracted answer.
- **Best-of-n** selects one candidate with a verifier, ranker, or judge — it can be correct even when the first sampled output was wrong.

Majority vote, best-of-n, self-consistency: not the same metric

- **Majority vote** scores the most frequent extracted answer.
- **Best-of-n** selects one candidate with a verifier, ranker, or judge — it can be correct even when the first sampled output was wrong.
- **Self-consistency** samples multiple reasoning paths and selects by answer agreement.

Majority vote, best-of-n, self-consistency: not the same metric

- **Majority vote** scores the most frequent extracted answer.
- **Best-of-n** selects one candidate with a verifier, ranker, or judge — it can be correct even when the first sampled output was wrong.
- **Self-consistency** samples multiple reasoning paths and selects by answer agreement.

Validation Rule (Comparable sampling)
A report must not compare a single-sample score, a pass@k estimate, a majority-vote score, a best-of-n score, and a self-consistency score as if they were the same metric.

Cost, latency, and what a number is allowed to claim

Every generated sample counts

Five items, two samples each: 10 calls, 300 prompt + 700 completion tokens per call = 10,000 tokens total.

- Best-of-2 solves four items: quality $4/5 = 0.8$.

Every generated sample counts

Five items, two samples each: 10 calls, 300 prompt + 700 completion tokens per call $= 10{,}000$ tokens total.

- Best-of-2 solves four items: quality $4/5 = 0.8$.
- Token-normalized: $4/(10{,}000/10^6) = 400$ solved items per million tokens.

Every generated sample counts

Five items, two samples each: 10 calls, 300 prompt + 700 completion tokens per call = 10,000 tokens total.

- Best-of-2 solves four items: quality $4/5 = 0.8$.
- Token-normalized: $4/(10{,}000/10^6) = 400$ solved items per million tokens.
- Toy pricing (\$0.20/\$0.80 per million prompt/completion tokens): \$0.0062 per run, $\approx$645 solved items/\$.

Every generated sample counts

Five items, two samples each: 10 calls, 300 prompt + 700 completion tokens per call = 10,000 tokens total.

- Best-of-2 solves four items: quality $4/5 = 0.8$.
- Token-normalized: $4/(10{,}000/10^6) = 400$ solved items per million tokens.
- Toy pricing ($0.20/$0.80 per million prompt/completion tokens): $0.0062 per run, ≈645 solved items/$.

The H200 packet's own ledger:

Work	Calls	Tokens
Candidate gen.	80	32,164
Judge scoring	120	77,752
Combined	200	109,916

[OBSERVED: Panther H200 32B packet, 2026-07-31]

Every generated sample counts

Five items, two samples each: 10 calls, 300 prompt + 700 completion tokens per call = 10,000 tokens total.

- Best-of-2 solves four items: quality $4/5 = 0.8$.
- Token-normalized: $4/(10{,}000/10^6) = 400$ solved items per million tokens.
- Toy pricing ($0.20/$0.80 per million prompt/completion tokens): $0.0062 per run, ≈645 solved items/$.

The H200 packet's own ledger:

Work	Calls	Tokens
Candidate gen.	80	32,164
Judge scoring	120	77,752
Combined	200	109,916

[OBSERVED: Panther H200 32B packet, 2026-07-31]

These are teaching values and one honest local ledger, not provider-price claims — every sampled answer is charged, including the ones later discarded.

Leakage, aging, and robustness

Leakage is a protocol risk, not a casual accusation

Definition (Leakage)
An overlap or dependency path between evaluation items and any process that can adapt the model, prompt, scorer, or evaluator to those items.

- Public solutions, prompt tuning on the target split, synthetic data seeded by benchmark items, and judge calibration on near-duplicates are four distinct contamination paths.

Leakage is a protocol risk, not a casual accusation

Definition (Leakage)
An overlap or dependency path between evaluation items and any process that can adapt the model, prompt, scorer, or evaluator to those items.

- Public solutions, prompt tuning on the target split, synthetic data seeded by benchmark items, and judge calibration on near-duplicates are four distinct contamination paths.
- **Benchmark aging**: a public benchmark loses discriminative value as it is repeatedly reported, reverse engineered, or optimized against — it does not become useless, but it changes what the score can prove.

Leakage is a protocol risk, not a casual accusation

Definition (Leakage)
An overlap or dependency path between evaluation items and any process that can adapt the model, prompt, scorer, or evaluator to those items.

- Public solutions, prompt tuning on the target split, synthetic data seeded by benchmark items, and judge calibration on near-duplicates are four distinct contamination paths.
- **Benchmark aging**: a public benchmark loses discriminative value as it is repeatedly reported, reverse engineered, or optimized against — it does not become useless, but it changes what the score can prove.

The correct response is to record data status, benchmark age, and mitigation — not to pretend a score has no history.

Prompt sensitivity has its own definition

$$\Delta_{\mathcal{T}} = \max_{\tau \in \mathcal{T}} M(\tau) - \min_{\tau \in \mathcal{T}} M(\tau)$$

- If terse, format-constrained, and few-shot prompts solve 14, 16, and 13 of 20 items, then $\Delta_{\mathcal{T}} = 0.15$.

Prompt sensitivity has its own definition

$$\Delta_{\mathcal{T}} = \max_{\tau \in \mathcal{T}} M(\tau) - \min_{\tau \in \mathcal{T}} M(\tau)$$

- If terse, format-constrained, and few-shot prompts solve 14, 16, and 13 of 20 items, then $\Delta_{\mathcal{T}} = 0.15$.
- A reviewed RTX 5090 diagnostic ran three templates, nine seeds, and 20 constructed integer items: 540/540 completed rows, 64 invalid parses (all at the 512-token cap), 207,347 total tokens.

Prompt sensitivity has its own definition

$$\Delta_{\mathcal{T}} = \max_{\tau \in \mathcal{T}} M(\tau) - \min_{\tau \in \mathcal{T}} M(\tau)$$

- If terse, format-constrained, and few-shot prompts solve 14, 16, and 13 of 20 items, then $\Delta_{\mathcal{T}} = 0.15$.
- A reviewed RTX 5090 diagnostic ran three templates, nine seeds, and 20 constructed integer items: 540/540 completed rows, 64 invalid parses (all at the 512-token cap), 207,347 total tokens.

[OBSERVED: RTX 5090 prompt-family and seed small set, 2026-07-21/22/24] — prompt-sensitivity evidence for the constructed set, not benchmark-capability evidence.

Judges, harness design, and the worked AIME harness

Rule graders and judges answer different questions

- **Rule scorer**: a grade through a specified, versioned procedure — reproducible, but reproducibility does not guarantee validity.

Gold class	Judge correct	Judge incorrect	Total
Correct	95	0	95

Rule graders and judges answer different questions

- **Rule scorer**: a grade through a specified, versioned procedure — reproducible, but reproducibility does not guarantee validity.
- **Judge model**: scores or ranks outputs inside the procedure; useful when exact-match rules miss helpfulness or clarity, but can prefer verbosity or share model biases.

Gold class	Judge correct	Judge incorrect	Total
Correct	95	0	95

Rule graders and judges answer different questions

- **Rule scorer**: a grade through a specified, versioned procedure — reproducible, but reproducibility does not guarantee validity.
- **Judge model**: scores or ranks outputs inside the procedure; useful when exact-match rules miss helpfulness or clarity, but can prefer verbosity or share model biases.

Gold class	Judge correct	Judge incorrect	Total
Correct	95	0	95
Incorrect	25	0	25

Rule graders and judges answer different questions

- **Rule scorer**: a grade through a specified, versioned procedure — reproducible, but reproducibility does not guarantee validity.
- **Judge model**: scores or ranks outputs inside the procedure; useful when exact-match rules miss helpfulness or clarity, but can prefer verbosity or share model biases.

Gold class	Judge correct	Judge incorrect	Total
Correct	95	0	95
Incorrect	25	0	25

Same-checkpoint judge: 79.17% accuracy, but 100% sensitivity and 0% specificity — it never once said `INCORRECT`. [OBSERVED: Panther H200 judge calibration, 2026-07-31]

From concept to code: the extractor that decides invalid

The extraction policy from the boxed-answer example above has a literal, book-owned home.

from `code/chapter_evaluation_methodology/strict_boxed_answer_checkpoint.py`

```python
def extract_one_boxed_integer(text: str) -> tuple[str, int | None]:
    boxes = findall(r"\boxed\{([^{}]*)\}", text)
    if not boxes:
        return "invalid:no-box", None
    if len(boxes) != 1:
        return "invalid:multiple-boxes", None
    payload = boxes[0].strip()
    if fullmatch(r"[+-]?\d+", payload) is None:
        return "invalid:non-integer", None
    return "ok", int(payload)
```

From concept to code: the extractor that decides invalid

The extraction policy from the boxed-answer example above has a literal, book-owned home.

from **code/chapter_evaluation_methodology/strict_boxed_answer_checkpoint.py**

```python
def extract_one_boxed_integer(text: str) -> tuple[str, int | None]:
    boxes = findall(r"\boxed\{([^{}]*)\}", text)
    if not boxes:
        return "invalid:no-box", None
    if len(boxes) != 1:
        return "invalid:multiple-boxes", None
    payload = boxes[0].strip()
    if fullmatch(r"[+-]?\d+", payload) is None:
        return "invalid:non-integer", None
    return "ok", int(payload)
```

Three named invalid codes, checked before a value is ever compared to the key — a parser failure can never silently become a model failure or a model success. [INSPECTED: code/chapter_evaluation_methodology/strict_boxed_answer_checkpoint.py]

The worked harness, assembled

Component	Executable choice
Items	Fixed, hashed AIME 2024 list; store id, hash, expected answer
Prompt	Versioned template requesting reasoning + one boxed integer
Invocation	R1 revision, 32,768 tokens, temp. 0.6, top-$p = 0.95$, 64 samples

The worked harness, assembled

Component	Executable choice
Items	Fixed, hashed AIME 2024 list; store id, hash, expected answer
Prompt	Versioned template requesting reasoning + one boxed integer
Invocation	R1 revision, 32,768 tokens, temp. 0.6, top-$p = 0.95$, 64 samples
Extraction	Exactly one boxed integer, else a named invalid code
Scoring	Exact-integer comparison; store binary label + diagnostics
Aggregation	Declared pass@1 estimator, denominator, invalids, item unit

The worked harness, assembled

Component	Executable choice
Items	Fixed, hashed AIME 2024 list; store id, hash, expected answer
Prompt	Versioned template requesting reasoning + one boxed integer
Invocation	R1 revision, 32,768 tokens, temp. 0.6, top-$p = 0.95$, 64 samples
Extraction	Exactly one boxed integer, else a named invalid code
Scoring	Exact-integer comparison; store binary label + diagnostics
Aggregation	Declared pass@1 estimator, denominator, invalids, item unit

The reported 79.8 row is a comparison point, not a value to copy into the local result column — a local row exists only after this pipeline runs end to end.

Concept-to-code map, and what the appendix does not claim

- **boxed-integer extractor** → `code/.../strict_boxed_answer_checkpoint.py`

Concept-to-code map, and what the appendix does not claim

- **boxed-integer extractor** → `code/.../strict_boxed_answer_checkpoint.py`
- **invocation record: templates, sampling, seed policy** → `experiments/.../protocol_v1.json`

Concept-to-code map, and what the appendix does not claim

- **boxed-integer extractor** → `code/.../strict_boxed_answer_checkpoint.py`
- **invocation record: templates, sampling, seed policy** → `experiments/.../protocol_v1.json`
- **benchmark item schema** → `experiments/.../math_exact_20_v1.json`
- **cost/latency & judge calibration** → `experiments/.../panther_h200_evaluation_32b_2026_07_31.json`

Concept-to-code map, and what the appendix does not claim

- **boxed-integer extractor** → `code/.../strict_boxed_answer_checkpoint.py`
- **invocation record: templates, sampling, seed policy** → `experiments/.../protocol_v1.json`
- **benchmark item schema** → `experiments/.../math_exact_20_v1.json`
- **cost/latency & judge calibration** → `experiments/.../panther_h200_evaluation_32b_2026_07_31.json`
- **executable task/prompt/model/metric surfaces** → `https://github.com/EleutherAI/lm-evaluation-harness`

Concept-to-code map, and what the appendix does not claim

- **boxed-integer extractor** → `code/.../strict_boxed_answer_checkpoint.py`
- **invocation record: templates, sampling, seed policy** → `experiments/.../protocol_v1.json`
- **benchmark item schema** → `experiments/.../math_exact_20_v1.json`
- **cost/latency & judge calibration** → `experiments/.../panther_h200_evaluation_32b_2026_07_31.json`
- **executable task/prompt/model/metric surfaces** → `https://github.com/EleutherAI/lm-evaluation-harness`

Record every stage, or the claim is description, not reproduction.

Left open: exact 671B DeepSeek-R1 AIME reproduction, independent blinded judge-agreement evidence, and any dollar or energy measurement. Chapter 2 owns the protocol's definitions; this appendix owns their executable realization.