

Lab 6.1 Design a Controlled Experiment

Gavin · AI-assisted course design

Contents

1 Goal	3
2 Part 1 — Make a falsifiable plan before calculating	3
2.1 Record the separate fixture prediction	4
3 Part 2 — Save the exact original fixture	4
3.1 Calculate before coding	5
4 Part 3 — Declare the uncertainty calculation	5
5 Part 4 — Implement the bounded reference calculation	6
5.1 Deliberately retain a rejected run	10
6 Part 5 — Diagnose pseudoreplication	10
7 Part 6 — Repair a confounded comparison	11
8 Part 7 — Interpret without moving the goalposts	11
9 Part 8 — Check the exact reference values	12
10 Optional — Audit a prior Lab without new model calls	13
11 Completion checklist	13
12 More Learning	13

1 Goal

Turn a vague claim into a testable plan, analyze an exact paired fixture, and explain two ways an attractive result can mislead: counting dependent rows as independent evidence and comparing conditions on different task mixtures.

Prerequisite: Lesson 6.1. Earlier model Labs are useful context but are not required to complete this Lab.

Time: about 75–100 minutes for planning, arithmetic, implementation, and interpretation.

Requirements: an existing Python 3 environment with its standard library. No GPU, model, package installation, credentials, account, inference API, network call, paid compute, or dataset download is required. The reference program uses only standard-library modules and ordinary local files.

Expected artifact: `experiment-plan.md`, `predictions.md`, the exact fixture CSV, `analyze_design.py`, the successful run directory, one deliberately rejected malformed-input run directory, and `submission.md`. Keep the original records and both outcomes.

 These scores are invented

Every score and count supplied below is an original synthetic teaching fixture. No LLM produced them. The calculations teach design and analysis; they establish nothing about actual prompt quality or model capability. The answer is visible in the lesson, so recording a prediction here rehearses a good habit rather than proving an analysis was genuinely blind.

The core consists of Parts 1–8. An optional audit of previously saved Lab records requires no new model execution and cannot replace the core exercise.

2 Part 1 — Make a falsifiable plan before calculating

Start from this deliberately vague claim:

Asking a model to check its answer improves its reasoning.

Write a hypothetical small instructional experiment in `experiment-plan.md`. You will not execute model calls in this Lab. Specify the following in enough detail that another learner could tell whether a proposed run followed your plan:

1. **Question and target:** the behavior, task population, and scope of the conclusion. Prefer a narrow claim about an answer score over an undefined claim about “reasoning.”
2. **Conditions:** the exact control and treatment instructions. Identify the changed text and which checkpoint, tokenizer, templates, context, decoding settings, and harness behavior would remain fixed.
3. **Primary outcome:** a scoring rule, its unit, and its denominator. State how incorrect, malformed, missing, refused, or timed-out outputs would be recorded. Distinguish an incorrect answer from a failure to obtain a measurement.

4. **Practical threshold:** the smallest gain that would justify the change, together with an acceptable resource cost. Explain why that threshold answers your question.
5. **Units and pairing:** the independent task families, related wording variants, repeat generations if any, and the unit used for uncertainty. Explain how the same tasks reach both conditions.
6. **Separation:** which families support development or validation and which would be untouched until final evaluation. Keep related variants together when the target is new families.
7. **Controls and confounders:** two plausible alternative explanations and one design choice addressing each. Explain what remains uncontrolled.
8. **Analysis and stopping:** a primary contrast, number of independent units, exclusions, uncertainty method and assumptions, and fixed resource limits. Mark secondary investigations as exploratory.
9. **Falsification:** a plausible outcome that would count against your claim and an outcome that would leave it unresolved.
10. **Record and governance:** the manifests, raw outputs, failures, and software versions you would retain; the data you are allowed to use; and any human-evaluation permissions needed.

Do not propose collecting private conversations, running untrusted code, acquiring a larger model, or starting paid resources as an automatic step. A public or synthetic fixture is enough for an instructional plan. If a future experiment needs network access or spending, specify it as a separate decision.

A fixed checkpoint identifier is appropriate if you already have one from an earlier Lab. If you do not, label it “not yet selected; model execution blocked until selected and verified” rather than inventing a model setup. The plan can be complete as a design exercise while remaining unexecuted.

2.1 Record the separate fixture prediction

Before copying the analysis program or consulting the solutions, create `predictions.md`. Record:

- whether you expect the synthetic treatment’s mean family effect to be positive;
- whether you expect it to establish a gain of at least **five rubric points**;
- what you expect to happen to a naive standard error when records are duplicated;
- whether a higher pooled score must imply a higher score within difficulty groups;
- whether you have already read the lesson’s numerical answer;
- one observation that would change your initial interpretation.

Date this file. Keep it unchanged afterward; add later reflections to `submission.md`. Prior exposure is not a failure. Concealing it would defeat the exercise.

3 Part 2 — Save the exact original fixture

Save the following as `synthetic_scores.csv`, with UTF-8 encoding and the header exactly as shown. Values are dimensionless **rubric points on an invented 0–100 scale**, not accuracy percentages. A and B identify two related wording variants within each family. There are no hidden model outputs behind the numbers.

```
family,template,control,treatment
F01,A,49,52
F01,B,51,56
F02,A,59,60
```

F02, B, 61, 64
 F03, A, 54, 59
 F03, B, 56, 63
 F04, A, 64, 63
 F04, B, 66, 67
 F05, A, 69, 72
 F05, B, 71, 76
 F06, A, 79, 76
 F06, B, 81, 80
 F07, A, 74, 81
 F07, B, 76, 85
 F08, A, 84, 85
 F08, B, 86, 89

For this exercise, the intended hypothetical sampling model has eight independent, identically distributed families. The two variants within a family are related measurements. Both control and treatment are applied to each variant. Condition application is at the wording/run level; the hypothetical independent sampling unit and the analysis cluster are the family. The primary analysis uses one average paired difference per family and gives each family equal weight.

This hypothetical independence is an assumption for demonstrating a calculation. The hand-authored fixture was not sampled from an actual task population. It cannot empirically justify that model, normality, or generalization to real LLM tasks.

Do not remove F06, which has a negative average change, or F04, which has zero average change. Neither is a corrupted observation. The primary statistic includes all eight families.

3.1 Calculate before coding

First read and record the fixed uncertainty specification in Part 3. Then complete the arithmetic below, keeping the prediction and decision rule unchanged.

For family i and variant $j \in \{A, B\}$, define

$$d_{ij} = y_{ij,T} - y_{ij,C}, \quad \bar{d}_i = \frac{d_{iA} + d_{iB}}{2}, \quad \bar{d} = \frac{1}{8} \sum_{i=1}^8 \bar{d}_i.$$

1. Calculate every row's paired difference and every family's mean difference.
2. Calculate the equally weighted family means for control and treatment.
3. Calculate $\bar{d}$ and the sample variance of the eight family differences, using denominator $8 - 1$.
4. Explain why the analysis sample size is eight, despite sixteen rows and thirty-two condition scores.
5. State the descriptive result separately from any claim about a hypothetical population.

4 Part 3 — Declare the uncertainty calculation

The reference uses one two-sided, family-level Student- t interval:

$$s_d^2 = \frac{\sum_i (\bar{d}_i - \bar{d})^2}{7}, \quad \text{SE} = \sqrt{\frac{s_d^2}{8}}, \quad I = \bar{d} \pm 2.365 \text{ SE}.$$

Here 2.365 is the three-decimal critical value for seven degrees of freedom and a two-sided 95% interval from the NIST table. A program using a more precise quantile will differ slightly at later decimal places; this Lab deliberately uses the displayed constant.

The theoretical exact 95% coverage result uses the exact critical quantile and requires independent, identically distributed normally distributed family differences. Using the displayed rounded value of 2.365 makes that nominal coverage an approximation, even under those assumptions. With only eight families, do not rely on a large-sample argument to ignore that assumption. For this authored fixture, report the interval as an **illustrative model-based calculation**, not calibrated empirical uncertainty about LLMs. If the eight specified families are the entire target, their displayed mean is known exactly and needs no sampling interval.

The primary interpretation rule is fixed before analysis: establishing the hypothetical five-point requirement would require the lower endpoint of this chosen interval to exceed five, along with acceptable measurement checks and the separately declared resource criteria. This is a conservative interval-based decision convention, not a complete deployment policy. Do not switch to a different tail or threshold after inspecting the result.

No bootstrap, randomization test, seed search, or p -value calculation is required. There is no stochastic procedure in the reference program, so the reproducibility field is “randomness: none,” rather than an irrelevant seed. If you later choose a resampling analysis, write a new exploratory plan, preserve whole family pairs, specify assumptions and bounds, and do not relabel it as this Lab’s prespecified analysis.

5 Part 4 — Implement the bounded reference calculation

Save the following as `analyze_design.py`. Read it before running. It performs exact rational arithmetic for means and variances, followed by floating-point square roots and interval endpoints. It also prints two deliberately invalid independence calculations, explicitly labeled as such.

The run refuses to overwrite an existing output directory. It copies the input and predictions, records their hashes and its own source hash, and retains a failure file if validation fails. The recorded timestamp shows when this program ran; it is not proof that the predictions were written without prior knowledge.

```
import argparse
import csv
import hashlib
import io
import json
import math
import platform
import sys
from datetime import datetime, timezone
from fractions import Fraction
from pathlib import Path
```

```

def sha256(data):
    return hashlib.sha256(data).hexdigest()

def average(values):
    if not values:
        raise ValueError("Cannot average an empty collection")
    return sum(values, Fraction(0)) / len(values)

def sample_variance(values):
    if len(values) < 2:
        raise ValueError("Sample variance needs at least two values")
    center = average(values)
    return sum((x - center) ** 2 for x in values) / (len(values) - 1)

def exact_and_decimal(value):
    return {"exact": str(value), "decimal": float(value)}

def write_json(path, value):
    path.write_text(json.dumps(value, indent=2, sort_keys=True, allow_nan=False) +
                    "\n",
                    encoding="utf-8")

def load_fixture(raw):
    reader = csv.DictReader(io.StringIO(raw.decode("utf-8")))
    if reader.fieldnames != ["family", "template", "control", "treatment"]:
        raise ValueError("Unexpected CSV header")
    rows = list(reader)
    expected = [(f"F{i:02d}", t) for i in range(1, 9) for t in ("A", "B")]
    actual = [(r["family"], r["template"]) for r in rows]
    if actual != expected:
        raise ValueError("Require exactly F01-A through F08-B in fixed order")
    clean = []
    for row in rows:
        if None in row or any(value is None for value in row.values()):
            raise ValueError("Missing or extra CSV fields")
        scores = [int(row[key]) for key in ("control", "treatment")]
        if any(not 0 <= value <= 100 for value in scores):
            raise ValueError("A score is outside the declared 0-100 scale")
        clean.append({"family": row["family"], "template": row["template"],
                      "control": scores[0], "treatment": scores[1]})
    return clean

```

```

def analyze(rows):
    families = []
    row_differences = []
    for index in range(0, 16, 2):
        pair = rows[index:index + 2]
        c = average([Fraction(r["control"]) for r in pair])
        t = average([Fraction(r["treatment"]) for r in pair])
        differences = [Fraction(r["treatment"] - r["control"]) for r in pair]
        row_differences.extend(differences)
        families.append({"family": pair[0]["family"], "control": c,
                        "treatment": t, "difference": average(differences)})
    differences = [f["difference"] for f in families]
    effect = average(differences)
    variance = sample_variance(differences)
    se = math.sqrt(float(variance / len(differences)))
    critical = 2.365
    low, high = float(effect) - critical * se, float(effect) + critical * se
    # Intentional counterexamples: these assumptions are invalid here.
    naive_row_se = math.sqrt(float(sample_variance(row_differences) / 16))
    duplicated = differences * 10
    naive_duplicate_se = math.sqrt(float(sample_variance(duplicated) / 80))
    serial_families = [{key: exact_and_decimal(value) if isinstance(value,
Fraction)
                        else value for key, value in f.items()} for f in families]

    return {
        "data_kind": "original synthetic teaching fixture; no model outputs",
        "rows": len(rows), "independent_family_count_under_model": 8,
        "families": serial_families,
        "control_mean": exact_and_decimal(average([f["control"] for f in
families])),
        "treatment_mean": exact_and_decimal(average([f["treatment"] for f in
families])),
        "paired_effect": exact_and_decimal(effect),
        "family_difference_sample_variance": exact_and_decimal(variance),
        "family_difference_sample_sd": math.sqrt(float(variance)),
        "family_mean_standard_error_under_model": se,
        "illustrative_t95_interval": [low, high], "critical_value_rounded":
critical,
        "interval_assumptions": "iid normal family differences; not established by
fixture",
        "interval_lower_endpoint_exceeds_five": low > 5,
        "positive_zero_negative_family_counts": [sum(x > 0 for x in differences),
sum(x == 0 for x in differences),
sum(x < 0 for x in differences)],
        "INVALID_naive_row_independence_se": naive_row_se,
        "INVALID_ten_copies_as_independent_se": naive_duplicate_se,
        "correct_se_after_exact_duplication": se,

```

```

}

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--data", required=True, type=Path)
    parser.add_argument("--plan", required=True, type=Path)
    parser.add_argument("--out", required=True, type=Path)
    args = parser.parse_args()
    args.out.mkdir(parents=True, exist_ok=False)
    manifest = {"started_utc": datetime.now(timezone.utc).isoformat(),
                "python": sys.version, "platform": platform.platform(),
                "randomness": "none", "network_required": False,
                "source_sha256": sha256(Path(__file__).read_bytes()),
                "status": "started"}
    write_json(args.out / "manifest.json", manifest)
    try:
        with args.data.open("rb") as handle:
            raw = handle.read(100_001)
        with args.plan.open("rb") as handle:
            plan = handle.read(100_001)
        if len(raw) > 100_000 or len(plan) > 100_000:
            raise ValueError("Input or prediction file exceeds 100 kB bound")
        if not plan.strip():
            raise ValueError("Predictions must exist before analysis")
        (args.out / "input.csv").write_bytes(raw)
        (args.out / "predictions.md").write_bytes(plan)
        manifest.update(data_sha256=sha256(raw), predictions_sha256=sha256(plan))
        report = analyze(load_fixture(raw))
        write_json(args.out / "report.json", report)
        manifest["status"] = "completed"
        print(json.dumps(report, indent=2, sort_keys=True, allow_nan=False))
    except Exception as error:
        manifest["status"] = "failed_validation_or_analysis"
        write_json(args.out / "failure.json",
                    {"type": type(error).__name__, "message": str(error)})
        raise
    finally:
        manifest["finished_utc"] = datetime.now(timezone.utc).isoformat()
        write_json(args.out / "manifest.json", manifest)

if __name__ == "__main__":
    main()

```

Run one valid analysis:

```
python3 analyze_design.py --data synthetic_scores.csv --plan predictions.md --out
run-original
```

Set a one-minute wall-time ceiling per invocation and a **1 MB total output ceiling** for the two reference runs. These are learner-enforced stop rules, not runtime promises: this small reference program does not implement a watchdog timer or a total-output quota. There are no unbounded iterations or external requests. Stop and inspect an unexpected hang instead of launching repeated copies. Keep both input files below 100 kB; the actual fixture is much smaller.

The program validates structure and scale, not scientific honesty: its hash identifies exactly which bytes you used, but it cannot determine whether you selected favorable scores. Compare your CSV against the displayed fixture before calling it the reference dataset. After running, inspect the manifest and every family record, not just the headline effect. The field `interval_lower_endpoint_exceeds_five` checks only the numerical endpoint rule; it does not assess measurement validity, resource limits, or the full practical decision.

5.1 Deliberately retain a rejected run

Make a separate copy named `malformed_scores.csv`. Change only the final row's template from B to A, creating a duplicate F08-A and a missing F08-B. Keep the original CSV unchanged. Run once into `run-rejected` using the same predictions file.

The program should stop with a nonzero exit status, record a failed status and `failure.json`, retain the malformed input and predictions, and produce no `report.json`. The expected error mentions the required fixed family/template order. If it instead produces an effect estimate, repair the validator before trusting any analysis.

This is a validation failure, not a negative treatment effect. Both deserve preservation, but they have different meanings. Do not include the malformed run as another replicate or replace its missing score with zero. If another bug appears, retain its failed directory and document the repair before a clearly named rerun.

6 Part 5 — Diagnose pseudoreplication

Compare three quantities in the successful report:

- the family-level standard error under the declared model;
- the deliberately invalid standard error treating sixteen related variants as independent;
- the deliberately invalid standard error treating ten exact copies of each family effect as eighty independent effects.

The means remain unchanged. The naive uncertainty shrinks even though no new families were observed. That is the failure: nominal sample size increased without the corresponding independent information.

Explain why the correct eight-family calculation must remain unchanged after exact duplication. State what new data would help answer a question about unfamiliar task families. More seeds for existing families may help characterize generation variability, but they do not automatically substitute for new families or new independently trained models.

Do not use the invalid fields as alternative confidence intervals. They are counterexamples with intentionally false assumptions. The report names them INVALID so they cannot reasonably be mistaken for the primary result.

7 Part 6 — Repair a confounded comparison

The following is a second, separate synthetic fixture. Each row gives a group count and its fixed mean score. It is not an extension of the paired dataset, and the counts are not extra replicates for its interval.

Condition	Difficulty group	Count	Mean rubric score
Control	Easy	1	80
Control	Hard	9	30
Treatment	Easy	9	70
Treatment	Hard	1	20

Before computing, predict the sign of the pooled treatment-minus-control contrast and the sign within each difficulty group. Then calculate:

1. Each condition's count-weighted pooled mean.
2. The treatment-minus-control difference within each difficulty group.
3. Both condition means standardized to an explicitly chosen **50% easy, 50% hard** target mixture.
4. The standardized contrast for that target mixture.

All four calculations need only multiplication, addition, and division. Save the arithmetic in `submission.md` or a tiny separate standard-library script. Do not calculate a significance test: the fixture supplies group means and counts, not independently sampled raw outcomes or within-group variability.

Explain the apparent reversal. The treatment group contains many more easy cases. A pooled comparison mixes a condition difference with a task-composition difference. State why controlling this observed difficulty variable still would not, by itself, eliminate every possible confounder in real observational data. Propose a prospective repair: evaluate both conditions on the same frozen task families, keep the difficulty mixture fixed, and control or randomize relevant execution differences.

Changing the mixture is also changing the estimand. Equal easy/hard weights are a declared target for this exercise, not a universally correct description of deployment traffic.

8 Part 7 — Interpret without moving the goalposts

Write 250–400 words that answer:

- What did the eight synthetic families show descriptively?
- What assumptions make the displayed interval a population uncertainty procedure, and which are not established here?
- Did the chosen procedure establish the five-point requirement? Is that equivalent to proving the true effect is below five?
- Which family contradicted the predicted direction, and why did it remain included?

- How did your actual interpretation differ from your recorded prediction?
- What did the rejected run check, and why is it not statistical evidence against treatment?
- What did the confounded fixture teach that the paired fixture did not?
- What would you change in the design of a future, genuinely empirical Lab?

Use the phrase “synthetic fixture” somewhere in the conclusion. Do not write “the model improved,” “the intervention was proved,” or “there is a 95% probability the true effect lies here.” No model was run and the frequentist interval has a different interpretation.

Retain any negative or inconclusive conclusion. If you notice an appealing post hoc subgroup, label it exploratory and say what fresh evidence would be needed. Do not revise the original plan to make the subgroup appear prespecified.

9 Part 8 — Check the exact reference values

i Arithmetic and interpretation checks

The sixteen paired row differences are

[3, 5, 1, 3, 5, 7, -1, 1, 3, 5, -3, -1, 7, 9, 1, 3].

The eight family differences are [4, 2, 6, 0, 4, -2, 8, 2]. Family-level control and treatment means are 67.5 and 70.5. Their equally weighted mean paired effect is exactly 3 rubric points.

The sum of squared family deviations from three is 72. The sample variance is $72/7$, standard deviation approximately 3.207135, and standard error under the model approximately 1.133893. With the specified rounded critical value, the interval is approximately [0.318342, 5.681658].

There are six positive, one zero, and one negative family effects. The interval’s lower endpoint does not exceed five. The chosen practical requirement is not established; the calculation also does not establish that the population effect is below five. For the eight fixed families themselves, the descriptive effect is known to be three. None of these statements is an empirical result about an LLM.

The naive row variance is $160/15 = 32/3$, yielding the invalid row-independent standard error $\sqrt{2/3} \approx 0.816497$. Ten copies of every family difference yield an invalid standard error $\sqrt{9/79} \approx 0.337526$. The correct eight-family standard error remains 1.133893 because the copies provide no new information.

For the separate confounded fixture, control’s pooled mean is $(80 + 9 \cdot 30)/10 = 35$; treatment’s is $(9 \cdot 70 + 20)/10 = 65$. The pooled contrast is +30. Within both difficulty groups the contrast is -10. At equal mixture weights, control averages 55, treatment 45, and the standardized contrast is -10.

The malformed input must not produce a numerical report. Its saved failure is evidence about validation behavior, not treatment performance.

10 Optional — Audit a prior Lab without new model calls

If you already have saved records from Labs 4.2 or 5.1–5.4, select one completed comparison. Do not download models, rerun inference, change thresholds, or replace the original conclusion.

Identify its primary outcome, paired unit, related rows, discovery/evaluation boundary, exclusions, and failures. State which sources of variation were measured and which were held fixed. A small hand-selected prompt set may warrant a descriptive table rather than a population interval. Do not attach this Lab's t formula merely because you can calculate a mean.

For Lab 4.2, distinguish request trials from candidate samples. For Lab 5.1, distinguish prompt variants from held-out template families. For Lab 5.2, distinguish training seeds from independently generated test examples. For Lab 5.3, distinguish discovery prompts, held-out prompts, intervention arms, and baseline repeats. For Lab 5.4, distinguish discovery from held-out scenarios, clause-order variants within a scenario, and baseline repeats used for invariance checks.

Write one specific improvement for a future experiment and one claim the existing record does support. Keep exploratory reanalysis separate and retain the original files. This audit is complete even if its most useful conclusion is that broader uncertainty cannot be estimated from the saved design.

11 Completion checklist

- The hypothetical plan names a falsifiable claim, units, controls, primary metric, split, limits, and failure policy.
- Predictions were saved before your analysis, with prior exposure disclosed.
- The valid run reproduces the exact family means, effect, variance, and specified interval.
- All sixteen rows and all eight families remain present, including negative and zero effects.
- The malformed-input run is retained and produces no misleading summary.
- The two invalid standard errors are explained as pseudoreplication examples.
- The confounded mixture calculations and prospective repair are correct.
- The conclusion separates a known finite-fixture mean, a conditional uncertainty calculation, and unmeasured real-world performance.

12 More Learning

- NIST, Confidence Limits for the Mean: relate each symbol to the eight family differences, then identify the assumptions the fixture cannot validate.
- NIST, Randomized Block Designs: compare a task-family block with the nuisance factors in the examples.
- Dror et al., The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing: consider why the statistical procedure must follow the metric and sampling design.