Statistical Inference & Applied Statistics

A Complete 10-Chapter Maths for Programmers Course

Topics covered:

Sampling & confidence intervals · hypothesis testing & the t-test

A/B testing in practice · correlation, regression & causation

Bayesian inference & updating beliefs

Capstone: designing and analyzing one real experiment start to finish

Exercises: 30 hands-on exercises with worked solutions

Format: A4 · Dark-theme code examples

Philip Osztromok · Generated with Claude

Table of Contents

CHAPTER 1 OF 10

From Descriptive to Inferential: What This Course Adds

Statistical Inference & Applied Statistics

Chapter 1 · From Descriptive to Inferential: What This Course Adds

Probability & Statistics Fundamentals' own Chapter 1 opened with a direction: model known, predict the outcome. Its Chapter 9 flipped that briefly — summarize data you already have. This course goes one step further than either: given only a limited, incomplete *sample* of data, how confidently can you say anything about the full, unobserved reality behind it? That's inference, and it's the genuinely hard, genuinely useful part of applied statistics.

Descriptive vs. Inferential — A Direction Change

	PROBABILITY & STATISTICS FUNDAMENTALS	THIS COURSE
Direction	Known model → predicted outcomes; real data → a summary of *that* data	A limited sample → a confident, quantified claim about the true, unobserved population
Example question	"What's the mean of these 7 response times?" (Ch.9)	"Given these 7 response times, what can we honestly say the *true* average response time is, across every request that will ever happen?"
Core tools	Probability rules, distributions, expected value, descriptive statistics	Sampling distributions, confidence intervals, hypothesis tests, regression, Bayesian updating

Chapter 9's own worked example computed a mean and median from exactly seven sampled response times. It never asked the harder question this course opens with: how much should anyone actually *trust* a number computed from just seven data points? That's the entire subject of this course.

Five Concrete Connections to Real Work

INFERENTIAL TOPIC	WHERE IT ACTUALLY SHOWS UP
Sampling & sampling distributions (Ch.2)	Every metric ever computed from "the last 10,000 sessions" rather than every session that will ever occur — which is nearly always the real situation
Confidence intervals (Ch.3)	Reporting "conversion rate: 5.2% ± 0.4%" instead of a falsely precise single number that hides how much uncertainty the sample size actually leaves

INFERENTIAL TOPIC	WHERE IT ACTUALLY SHOWS UP
Hypothesis testing & A/B testing (Ch.4–6)	Deciding whether a new feature genuinely moved a metric, or whether the observed difference is just ordinary sample-to-sample noise
Correlation, causation & regression (Ch.7–8)	Analytics dashboards showing "X correlates with Y," and the classic mistake of assuming that means X causes Y; predicting a continuous outcome from real data
Bayesian updating (Ch.9)	Formally revising a belief as new evidence arrives over time — extending Probability & Statistics Fundamentals' own Bayes' Theorem chapter from a single fixed calculation into an ongoing, evidence-accumulating process

What This Course Builds Directly On

NOT STARTING FROM ZERO — TWO FORWARD REFERENCES, FINALLY PAID OFF

Probability & Statistics Fundamentals planted two explicit forward references into this course. Its Chapter 8 (the Central Limit Theorem) showed that a sample mean's own distribution approaches normal, with a shrinking standard error as sample size grows — exactly the machinery Chapter 2 of this course builds confidence intervals on top of. Its Chapter 4 (Bayes' Theorem) showed how to update a single probability given one piece of evidence — exactly the mechanism Chapter 9 of this course generalizes into an ongoing belief-updating process.

What This Course Won't Cover

A few genuinely related topics stay deliberately out of scope, to keep this course focused on the core inferential toolkit rather than sprawling into adjacent specialties:

- **Advanced experimental design** — multi-armed bandits, sequential testing, and other more sophisticated A/B testing variants beyond the fixed-sample-size approach Chapter 6 covers
- **Machine learning model evaluation** — cross-validation, precision/recall, and related ML-specific metrics belong to Data Science & ML's own `ds1` / `ml1` courses, not here
- **Full computational Bayesian methods** — MCMC sampling and similar techniques for genuinely complex Bayesian models stay out of scope; Chapter 9 covers Bayesian *reasoning*, not Bayesian computation at scale

WHY DRAW THE LINE HERE INSTEAD OF COVERING EVERYTHING AT ONCE

Each of those three areas is substantial enough to deserve its own real depth rather than a rushed final section. This course stays tightly scoped to the inferential reasoning every engineer or analyst needs directly — confidence, significance, correlation, and updating a belief — the concrete foundation those more specialized topics would each build on.

Where This Course Is Headed

CHAPTER	TOPIC
2	Sampling & the Sampling Distribution
3	Confidence Intervals
4	Hypothesis Testing Fundamentals
5	The t-test & Comparing Two Groups
6	A/B Testing in Practice
7	Correlation vs. Causation
8	Linear Regression as Statistical Inference
9	Bayesian Inference & Updating Beliefs
10	Capstone — Designing and Analyzing a Real Experiment

THIS COURSE'S THROUGHLINE

Every chapter answers a version of the same question: how much can a limited, incomplete sample actually tell you about the truth, and how do you act correctly under the uncertainty that remains? That's a genuinely different skill from Probability & Statistics Fundamentals' own forward reasoning — it's reasoning backward, from imperfect evidence to a defensible conclusion, which is exactly what's needed to trust an A/B test result, size a rollout confidently, or know when "the numbers look different" actually means something.

Hands-On Exercises

EXERCISE 1

Classify each of the following as belonging more to Probability & Statistics Fundamentals (a known model, or a plain summary of data in hand) or to this course (drawing a conclusion about an unobserved population from limited data): (a) computing the average of 50 measured response times, (b) claiming, from those same 50 measurements, that the true average across all future requests is likely between 190ms and 210ms, (c) given a known fair coin, finding the probability of 3 heads in 5 flips, (d) deciding whether a 2% lift in conversion rate seen in a 2-week experiment reflects a real effect or random noise.

View solution

EXERCISE 2

A colleague says "we don't need confidence intervals — we computed the mean directly from our data, so we already know the real answer." Using this chapter's own descriptive-vs-inferential distinction, explain what's wrong with this reasoning.

📄 View solution

EXERCISE 3

Explain, in your own words, how this course's Chapter 2 (sampling distributions) depends directly on Probability & Statistics Fundamentals' own Chapter 8 (the Central Limit Theorem), and how this course's Chapter 9 (Bayesian updating) depends directly on that same course's Chapter 4 (Bayes' Theorem) — using this chapter's own "not starting from zero" finding.

📄 View solution

CHAPTER 1 QUICK REFERENCE

- **Descriptive statistics** (Fundamentals Ch.9) summarizes data you already have; **inferential statistics** (this course) uses that data to draw a confident, quantified conclusion about the unobserved whole
- Five direct connections: sampling → any real metric from a subset of data, confidence intervals → honest uncertainty reporting, hypothesis/A-B testing → "is this difference real," correlation/regression → dashboards and prediction, Bayesian updating → revising beliefs over time
- Built directly on two forward references from the sibling course: the CLT (Ch.8 there) feeds sampling distributions here; Bayes' Theorem (Ch.4 there) feeds Bayesian updating here
- Deliberately out of scope: advanced experimental design, ML model evaluation, and full computational Bayesian methods
- **Next chapter:** Sampling and the sampling distribution

CHAPTER 2 OF 10

Sampling & the Sampling Distribution

Statistical Inference & Applied Statistics

Chapter 2 · Sampling & the Sampling Distribution

Chapter 1 promised this chapter would build directly on Probability & Statistics Fundamentals' own Central Limit Theorem. Here's the payoff: the CLT isn't just a curiosity about averages — it's the entire mathematical foundation for saying anything trustworthy about a limited sample.

Population vs. Sample

The **population** is the complete set of everything you'd ideally want to know about — every request that will ever be made, every user who will ever sign up. The **sample** is the limited subset actually collected and measured. Almost every real measurement in engineering is a sample, not the full population — you never have *every* request that will ever occur, only the ones logged so far.

POINT ESTIMATE

A **point estimate** is a single number, computed from a sample, used to estimate an unknown population parameter — a sample mean $\bar{x}$ estimating the true population mean μ, for instance.

Sampling Error — Not a Mistake, Just Randomness

Sampling error is the natural, expected difference between a point estimate and the true population value, arising purely from *which* subset happened to be sampled — not from anything done wrong. Two different samples of the same size, drawn from the exact same population, will almost always produce two slightly different sample means, purely by chance.

The Sampling Distribution

Imagine repeating the sampling process many times — draw a fresh sample of size n, compute its mean, and repeat. The distribution of *all those sample means* is the **sampling distribution of the sample mean** — a genuinely different object from the distribution of individual data points.

THIS IS EXACTLY PROBABILITY & STATISTICS FUNDAMENTALS CHAPTER 8'S OWN CLT

That course's own Central Limit Theorem already proved this sampling distribution approaches Normal(`μ` , `σ²/n`) as `n` grows — regardless of the shape of the original population. Its own standard error formula, `SE = σ/√n` , *is* the standard deviation of this sampling distribution. Nothing new needs deriving here — this chapter is that formula, put to direct use.

Worked Example: Response Times, Revisited

Reusing Probability & Statistics Fundamentals' own response-time model — true population mean `μ = 200ms` , true population standard deviation `σ = 30ms` (values a team would rarely know for certain in reality, but useful here to see sampling behavior clearly):

SAMPLE SIZE N	STANDARD ERROR (Σ/√N)
25	6.0ms
100	3.0ms
900	1.0ms

Suppose a team samples `n = 25` requests and gets a sample mean of `204ms` — 4ms above the true 200ms. Standardizing this as a z-score against the *sampling distribution* (not individual requests): `z = (204 − 200) / 6 = 0.667` . That's a thoroughly unremarkable result — well within a single standard error, exactly the kind of harmless wobble sampling error alone would produce. Nothing here suggests anything actually changed.

Sampling Error vs. Sampling Bias — A Critical Distinction

Collecting more data shrinks sampling error, per the `σ/√n` formula above — but only if the sampling *method* itself is sound. If the method itself systematically excludes part of the population, more data doesn't fix anything.

A REAL TRAP: SAMPLING BIAS SURVIVES ANY AMOUNT OF EXTRA DATA

Suppose a team only measures response times during daytime traffic, missing a slower nightly batch job that runs every night. No matter how many daytime samples they collect — a thousand, a million — their estimate of "true average response time across *all* traffic" stays systematically wrong, consistently too fast. More data makes a *biased* estimate more precisely wrong, not more accurate. Sampling error is fixed by collecting more data; sampling bias is fixed only by fixing the sampling method itself.

Sampling & the Sampling Distribution in Code

```python
import math, random

def standard_error(sigma, n):
    return sigma / math.sqrt(n)

print(standard_error(30, 25))   # 6.0
print(standard_error(30, 100))  # 3.0

# A quick simulation: draw many samples, see the sample-mean spread shrink
def simulate_sample_means(mu, sigma, n, num_samples=1000):
    return [
        sum(random.gauss(mu, sigma) for _ in range(n)) / n
        for _ in range(num_samples)
    ]

means_n25 = simulate_sample_means(200, 30, n=25)
means_n100 = simulate_sample_means(200, 30, n=100)
# The spread of means_n100 will be visibly tighter than means_n25 —
# run it and check with statistics.stdev() against the SE formula above
```

Hands-On Exercises

EXERCISE 1

A metric's true population standard deviation is `σ = 200` users. A sample of `n = 25` days gives a sample mean 90 users above the assumed population mean. Compute the standard error, then the z-score for this sample mean against the sampling distribution. Is this result unremarkable (well within 1–2 standard errors) or notably large?

View solution

EXERCISE 2

A metric has population standard deviation `σ = 80`. Compute the standard error for sample sizes `n = 4`, `n = 16`, and `n = 64`. Describe the pattern in how much `n` must grow to halve the standard error, connecting it back to Probability & Statistics Fundamentals' own Chapter 8 finding.

View solution

EXERCISE 3

A company estimates customer satisfaction using only responses from an optional online contact form — a small, self-selected fraction of all customers. Their estimated satisfaction score is very high. Using this chapter's own sampling-error-vs-sampling-bias distinction, explain why collecting 10 times more responses through that exact same method would not fix the underlying problem with this estimate.

View solution

CHAPTER 2 QUICK REFERENCE

- **Population** = everything you'd ideally want to know; **sample** = the limited subset actually collected
- **Point estimate:** a single sample statistic (e.g., $\bar{x}$) used to estimate an unknown population parameter (e.g., μ)
- **Sampling error** is the natural, expected sample-to-sample variation — not a mistake
- **Sampling distribution of the sample mean:** approaches Normal(μ, σ^2/n) per the CLT — $SE = \sigma/\sqrt{n}$ is its standard deviation
- **Sampling error** shrinks with more data; **sampling bias** (a systematically unrepresentative method) does not — more biased data is just more precisely wrong
- **Next chapter:** Confidence intervals

CHAPTER 3 OF 10

Confidence Intervals

Statistical Inference & Applied Statistics

Chapter 3 · Confidence Intervals

Chapter 2 established that a point estimate always carries sampling error. A confidence interval is the direct, practical fix: instead of reporting a single number as if it were exact, report a *range* of plausible values, together with how confident that range actually is.

Building a Confidence Interval

CONFIDENCE INTERVAL FORMULA

`CI = point estimate ± (critical value × standard error)`

The **critical value** (`z*`) comes straight from Probability & Statistics Fundamentals' own standard normal distribution — how many standard errors wide the interval needs to be to capture the stated percentage of the sampling distribution:

CONFIDENCE LEVEL	CRITICAL VALUE (Z*)
90%	1.645
95%	1.960
99%	2.576

Worked Example: The Response-Time Sample, Revisited

Reusing Chapter 2's own sample: `n = 25` requests, sample mean `x̄ = 204ms`, `σ = 30ms` (still assumed known here — Chapter 5's own t-test covers the more realistic case where `σ` itself must be estimated). Standard error: `SE = 30/√25 = 6ms`.

CONFIDENCE LEVEL	INTERVAL	WIDTH
90%	[194.13, 213.87]	19.74ms
95%	[192.24, 215.76]	23.52ms

CONFIDENCE LEVEL	INTERVAL	WIDTH
99%	[188.54, 219.46]	30.91ms

THE CONFIDENCE-VS-WIDTH TRADEOFF

Higher confidence needs a wider net — a 99% interval is genuinely wider than a 90% one, because casting a narrower net risks missing the true value more often. There's no free lunch: more certainty always costs precision, and the choice of confidence level is a real decision about which tradeoff a given situation actually needs.

Width Shrinks With Sample Size

Reusing Chapter 2's own three sample sizes, at a fixed 95% confidence level:

N	SE	95% CI WIDTH
25	6.0ms	23.52ms
100	3.0ms	11.76ms
900	1.0ms	3.92ms

More data doesn't change the confidence level — it tightens the interval at whatever confidence level was chosen, exactly Chapter 2's own `σ/√n` shrinkage, now made directly visible in the width of the reported range.

What a Confidence Interval Actually Means

THE SINGLE MOST COMMON MISINTERPRETATION IN ALL OF STATISTICS

A 95% confidence interval does **not** mean "there's a 95% probability the true mean falls in this specific interval." The true population mean is a fixed, though unknown, number — it either is or isn't in this particular interval, with no probability attached to that once the interval is already computed. The correct interpretation: **if this exact sampling-and-interval-building process were repeated many times**, about 95% of the resulting intervals would contain the true mean. The 95% describes the reliability of the *method* across repeated use, not the odds for this one specific interval.

A Forward Note: Proportions

Everything above used a sample *mean*. Chapter 6's A/B testing material needs confidence intervals for a *proportion* instead — a conversion rate, not an average response time. The idea is identical; only the standard

error formula changes, to `SE = √(p(1−p)/n)`. The confidence interval itself still follows the exact same `estimate ± z* × SE` shape.

Confidence Intervals in Code

```
Z_CRITICAL = {90: 1.645, 95: 1.96, 99: 2.576}

def confidence_interval(point_estimate, sigma, n, confidence=95):
    se = sigma / (n ** 0.5)
    z = Z_CRITICAL[confidence]
    margin = z * se
    return (point_estimate - margin, point_estimate + margin)

print(confidence_interval(204, sigma=30, n=25, confidence=95))
# (192.24, 215.76) — matches the worked example

print(confidence_interval(204, sigma=30, n=25, confidence=99))
# (188.54, 219.46) — wider, at higher confidence
```

Hands-On Exercises

EXERCISE 1

A sample of `n = 16` gives a sample mean of `550`, with a known `σ = 48`. Compute the standard error, then the 95% confidence interval.

View solution

EXERCISE 2

Given a fixed standard error of `SE = 10`, compute the width of the confidence interval at the 90%, 95%, and 99% confidence levels, using this chapter's own critical values. Confirm the widths increase as confidence increases, matching this chapter's own confidence-vs-width tradeoff.

View solution

EXERCISE 3

A teammate says "we're 95% confident the true average is between 190ms and 210ms, so there's a 95% chance the real value is in that range." Using this chapter's own correct interpretation, explain specifically what's wrong with the teammate's restatement, and give the statistically correct version of the claim.

View solution

CHAPTER 3 QUICK REFERENCE

- **CI formula:** `point estimate ± (z* × SE)`, reusing Chapter 2's own standard error directly
- **Critical values:** 90% → 1.645, 95% → 1.96, 99% → 2.576
- Higher confidence requires a wider interval — the confidence-vs-width tradeoff, with no way around it
- Interval width shrinks with sample size at any fixed confidence level, following Chapter 2's own `σ/√n` shrinkage exactly
- A 95% CI means: if this method were repeated many times, ~95% of the resulting intervals would contain the true value — **not** "95% probability this specific interval contains it"
- The same `estimate ± z* × SE` shape applies to proportions too, with `SE = √(p(1−p)/n)` — Chapter 6's own A/B testing material
- **Next chapter:** Hypothesis testing fundamentals

CHAPTER 4 OF 10

Hypothesis Testing Fundamentals

Statistical Inference & Applied Statistics

Chapter 4 · Hypothesis Testing Fundamentals

Chapter 2 ended on an unresolved tension: a z-score of 2.25 was called "notably large," but the chapter admitted it didn't yet have "the formal tool to make that determination with real statistical rigor." This chapter is that tool.

The Null and Alternative Hypothesis

TWO COMPETING CLAIMS

H_0 (null hypothesis): the default, "nothing changed" claim.
H_1 (alternative hypothesis): the claim that there genuinely is an effect or difference.

A hypothesis test never *proves* H_1 — it only asks whether the observed data provides strong enough evidence to **reject H_0**. Failing to reject H_0 isn't the same as proving it true; it just means the evidence wasn't strong enough to rule it out.

Significance Level — Directly Tied to Chapter 3's Confidence Level

The **significance level** (`α`) is the threshold risk of a false rejection a team is willing to accept — conventionally `α = 0.05` (5%). It's not a new idea: `α = 1 − confidence level`, exactly Chapter 3's own 95% confidence level restated from the opposite direction.

The Test Statistic and P-Value

Z-STATISTIC AND P-VALUE (KNOWN Σ)

`z = (x̄ − μ₀) / SE` `p-value = 2 × (1 − Φ(|z|))` for a two-tailed test, using the same standard normal CDF `Φ` from Probability & Statistics Fundamentals Chapter 8.

The p-value answers a precise question: *if H_0 were actually true*, how likely is a result at least this extreme, purely from sampling error? A small p-value means the observed data would be a genuinely unusual coincidence under H_0 — evidence favoring H_1 instead.

DECISION RULE

If `p-value < α`, reject H_0 ("statistically significant"). Otherwise, fail to reject H_0.

Resolving Chapter 2's Own Two Examples

Testing `H₀: μ = 200ms` against `H₁: μ ≠ 200ms`, reusing Chapter 2's own two scenarios:

SCENARIO	Z	P-VALUE	DECISION AT A=0.05
x̄=204ms, n=25, σ=30 (Ch.2's main example)	0.667	0.505	Fail to reject H_0 — confirms Ch.2's "unremarkable" call
90-unit shift, n=25, σ=200 (Ch.2's Exercise 1)	2.25	0.024	**Reject H_0** — formally confirms Ch.2's "notably large" call

Chapter 2 could only eyeball these results informally. This chapter turns that instinct into a precise, defensible number.

Type I and Type II Errors

Every hypothesis test can go wrong in two distinct ways:

	H_0 IS ACTUALLY TRUE	H_0 IS ACTUALLY FALSE
Reject H_0	**Type I error** (false positive) — rate controlled directly by α	Correct decision
Fail to reject H_0	Correct decision	**Type II error** (false negative) — a real effect gets missed

Lowering `α` reduces Type I errors directly, but — with a fixed sample size — tends to increase Type II errors, since a stricter threshold makes real effects harder to detect too. Neither error can be eliminated without more data.

Two Real Cautions

THE MULTIPLE-TESTING TRAP

At `α = 0.05`, testing 15 genuinely null hypotheses (no real effect in any of them) still produces, on average, `15 × 0.05 = 0.75` "statistically significant" results purely by chance. Running many tests and reporting only the "significant" ones — without correcting for how many tests were actually run — is a direct path to false conclusions, exactly the base-rate reasoning Probability & Statistics Fundamentals' own Bayes' Theorem chapter warned about from a different angle.

STATISTICAL SIGNIFICANCE ≠ PRACTICAL SIGNIFICANCE

With a large enough sample, even a genuinely tiny, meaningless difference can become "statistically significant" — a huge `n` shrinks the standard error enough that even trivial effects produce a small p-value. A significant result answers "is there probably a real effect," not "is this effect big enough to matter." Both questions need answering separately.

Hypothesis Testing in Code

```
import math

def standard_normal_cdf(z):
    return 0.5 * (1 + math.erf(z / math.sqrt(2)))

def z_test(sample_mean, mu0, sigma, n):
    se = sigma / math.sqrt(n)
    z = (sample_mean - mu0) / se
    p_value = 2 * (1 - standard_normal_cdf(abs(z)))
    return z, p_value

z, p = z_test(sample_mean=204, mu0=200, sigma=30, n=25)
print(z, p)  # 0.667, 0.505 — matches Chapter 2's own "unremarkable" result

alpha = 0.05
print("reject H0" if p < alpha else "fail to reject H0")
```

Hands-On Exercises

EXERCISE 1

Testing $H_0: \mu = 50$ against $H_1: \mu \neq 50$, a sample of $n = 36$ gives $\bar{x} = 54$, with known $\sigma = 12$. Compute the standard error, the z-statistic, and the p-value. State the decision at $\alpha = 0.05$.

View solution

EXERCISE 2

A team runs a hypothesis test and rejects H_0, concluding their new feature improved conversion. It later turns out the feature had no real effect at all — the observed difference was just sampling noise. Which type of error occurred: Type I or Type II? Now describe a second, different scenario where the opposite error type occurs instead.

View solution

EXERCISE 3

A team runs 20 independent A/B tests in a single quarter, all at `α = 0.05`, and none of the 20 features actually has any real effect. Using this chapter's own multiple-testing finding, compute the expected number of tests that will show a "statistically significant" result purely by chance, and explain what this implies for how the team should interpret a report claiming "3 out of 20 tests were significant this quarter."

View solution

CHAPTER 4 QUICK REFERENCE

- **H_0** (null, "no effect") vs **H_1** (alternative, "real effect") — a test can reject H_0, never prove H_1
- **Significance level α** = 1 − confidence level (Chapter 3) — conventionally 0.05
- **Z-statistic:** `z = (x̄ − μ₀)/SE`; **p-value:** `2(1 − Φ(|z|))` — reject H_0 when p-value < α
- **Type I error:** false positive (rejecting a true H_0), rate = α; **Type II error:** false negative (missing a real effect)
- Running many tests inflates the expected number of false positives purely by chance — the multiple-testing trap
- Statistical significance ("probably a real effect") is not the same question as practical significance ("is it big enough to matter")
- **Next chapter:** The t-test and comparing two groups

CHAPTER 5 OF 10

The t-test & Comparing Two Groups

Statistical Inference & Applied Statistics

Chapter 5 · The t-test & Comparing Two Groups

Every z-test in Chapter 4 quietly assumed the population standard deviation, `σ`, was already known. In real work, that's almost never true — only the *sample* standard deviation `s` is available, computed exactly the way Probability & Statistics Fundamentals' own Chapter 9 defined it, with Bessel's `n−1` correction. That single substitution changes the math more than it might seem.

Why Estimating σ Changes the Distribution

When `σ` is estimated from the same limited sample as `s`, the resulting test statistic carries extra uncertainty on top of ordinary sampling error — and its true sampling distribution is no longer exactly normal. It follows the **t-distribution** instead: shaped like the normal bell curve, but with **heavier tails**, reflecting that extra layer of uncertainty from estimating `s` itself.

DEGREES OF FREEDOM — DIRECTLY FROM CHAPTER 9'S OWN FORMULA

A one-sample t-test uses `df = n − 1` — exactly the denominator of the sample variance formula, Probability & Statistics Fundamentals Chapter 9's own Bessel's correction, reused here as the t-distribution's own shape parameter.

THE T-DISTRIBUTION CONVERGES TO NORMAL AS N GROWS

As `df` (and therefore `n`) grows, the t-distribution's tails thin out and it converges toward the standard normal — the extra uncertainty from estimating `s` matters less and less with more data, exactly the CLT's own "more data, more predictable" pattern from Chapter 8 of the sibling course, now applied to the act of estimating `σ` itself.

The One-Sample t-test

ONE-SAMPLE T-STATISTIC

`t = (x̄ − μ₀) / (s/√n)` — identical in shape to Chapter 4's z-statistic, but built from the sample standard deviation `s`, and compared against a t-distribution with `df = n − 1`, not the standard normal.

Worked example: a team doesn't know the true `σ`, only their sample — `n = 25`, `x̄ = 204ms`, `s = 32ms`. Testing `H₀: μ = 200`:

QUANTITY	VALUE
SE = s/√n	32/5 = 6.4ms
t	(204−200)/6.4 = 0.625
df	25 − 1 = 24
Critical value (two-tailed, α=0.05, df=24)	2.064
Decision	|0.625| < 2.064 → fail to reject H_0

Same conclusion as Chapter 4's own z-test on similar numbers — unsurprising, since a t-distribution with 24 degrees of freedom is already close to normal.

Comparing Two Groups: The Two-Sample t-test

Comparing two independent groups' means — exactly the shape of an A/B test — uses **Welch's t-test**, which doesn't assume the two groups share the same variance (a more honest default than assuming equal variances, and the version most statistical software now uses by default):

WELCH'S TWO-SAMPLE T-STATISTIC

$$t = (\bar{x}_1 - \bar{x}_2) / \sqrt{(s_1^2/n_1 + s_2^2/n_2)}$$

DEGREES OF FREEDOM FOR WELCH'S TEST — A DELIBERATE SIMPLIFICATION

Welch's test technically needs the Welch–Satterthwaite equation for its exact degrees of freedom — a genuinely messy formula. A common, more conservative simplification (used here) is $df = \min(n_1, n_2) - 1$, which tends to slightly understate the true df and therefore errs toward being more cautious about rejecting H_0 — a reasonable simplification for this course's own scope.

Worked example: Group A (control), $n_1=20$, $\bar{x}_1=205ms$, $s_1=25ms$; Group B (treatment), $n_2=22$, $\bar{x}_2=190ms$, $s_2=28ms$.

QUANTITY	VALUE
SE_diff = √(25²/20 + 28²/22)	≈ 8.178
t = (205−190)/8.178	≈ 1.834
df = min(20,22) − 1	19
Critical value (two-tailed, α=0.05, df=19)	2.093
Decision	|1.834| < 2.093 → fail to reject H_0

A VISIBLE DIFFERENCE ISN'T AUTOMATICALLY A SIGNIFICANT ONE

The treatment group's response time looks noticeably lower — 190ms vs 205ms, a real 15ms, roughly 7% relative drop. Yet the t-test can't rule out that this gap is just sampling noise at this sample size. This is exactly the gap Chapter 6's own A/B testing chapter exists to close properly — eyeballing a difference and formally testing it are genuinely different things.

The t-test in Code

```
import math

# Common two-tailed alpha=0.05 critical values, by degrees of freedom
T_CRITICAL_05 = {9: 2.262, 14: 2.145, 15: 2.131, 19: 2.093, 24: 2.064, 30: 2.042}

def one_sample_t(sample_mean, mu0, s, n):
    se = s / math.sqrt(n)
    t = (sample_mean - mu0) / se
    return t, n - 1

def welch_two_sample_t(mean1, s1, n1, mean2, s2, n2):
    se_diff = math.sqrt((s1**2 / n1) + (s2**2 / n2))
    t = (mean1 - mean2) / se_diff
    df = min(n1, n2) - 1  # conservative simplification
    return t, df

t, df = one_sample_t(204, mu0=200, s=32, n=25)
print(t, df)  # 0.625, 24

t2, df2 = welch_two_sample_t(205, 25, 20, 190, 28, 22)
print(t2, df2)  # 1.834..., 19
```

Hands-On Exercises

EXERCISE 1

Testing $H_0: \mu = 50$, a sample of $n = 16$ gives $\bar{x} = 54$, $s = 10$. Compute the standard error, the t-statistic, and the degrees of freedom. Using the critical value 2.131 (two-tailed, α=0.05, df=15), state the decision.

View solution

EXERCISE 2

Group 1: $n_1=15$, $\bar{x}_1=48$, $s_1=6$. Group 2: $n_2=18$, $\bar{x}_2=53$, $s_2=7$. Compute Welch's t-statistic, the conservative degrees of freedom, and using the critical value 2.145 (two-tailed, α=0.05, df=14), state the decision.

View solution

EXERCISE 3

Explain, in your own words, why the t-distribution needs heavier tails than the normal distribution when σ is unknown and estimated by s, and why a small-sample t-test's critical value (like 2.262 at df=9) is noticeably larger than the z-test's fixed 1.96. Then explain why that gap shrinks as sample size grows.

View solution

CHAPTER 5 QUICK REFERENCE

- When σ is unknown and estimated by sample s, use the **t-distribution** — heavier tails than normal, reflecting the extra uncertainty from estimating s itself
- **Degrees of freedom** (one-sample): $df = n - 1$ — the exact same denominator as Chapter 9's own Bessel's correction
- **One-sample t:** $t = (\bar{x} - \mu_0)/(s/\sqrt{n})$ — same shape as the z-statistic, compared against a t-distribution instead
- **Welch's two-sample t:** $t = (\bar{x}_1 - \bar{x}_2)/\sqrt{(s_1^2/n_1 + s_2^2/n_2)}$, with a conservative $df = \min(n_1, n_2) - 1$
- The t-distribution converges to the normal distribution as n (and therefore df) grows
- A visible descriptive difference between two groups doesn't automatically mean a statistically significant one — Chapter 6's own A/B testing chapter builds directly on this
- **Next chapter:** A/B testing in practice

CHAPTER 6 OF 10

A/B Testing in Practice

Statistical Inference & Applied Statistics

Chapter 6 · A/B Testing in Practice

Chapter 5's own visible-but-not-significant response-time gap set up the exact question this chapter answers properly. Real A/B tests almost always compare a **proportion** — a conversion rate, a click-through rate — not a continuous mean. That needs the proportion-specific version of everything built so far.

The Two-Proportion Z-Test

Comparing two independent conversion rates uses two *different* standard error formulas, depending on the question:

FOR THE HYPOTHESIS TEST (H_0: $P_1 = P_2$) — POOLED SE

$\hat{p} = (x_1+x_2)/(n_1+n_2)$ $SE_{pooled} = \sqrt{\hat{p}(1-\hat{p})(1/n_1 + 1/n_2)}$ $z = (p_1-p_2)/SE_{pooled}$

FOR THE CONFIDENCE INTERVAL ON THE DIFFERENCE — UNPOOLED SE

$SE_{diff} = \sqrt{p_1(1-p_1)/n_1 + p_2(1-p_2)/n_2}$ $CI = (p_1-p_2) \pm z^* \times SE_{diff}$

The distinction matters: the hypothesis test assumes H_0 is true (the two groups share one real rate, so it's honest to pool them into a single best estimate for that shared rate). The confidence interval makes no such assumption — it uses each group's own separate observed rate, since it's estimating how different the two groups genuinely are, not testing whether they're the same.

A VALIDITY CHECK WORTH DOING BEFORE TRUSTING THE NORMAL APPROXIMATION

This entire z-test relies on the normal approximation to the binomial distribution (Probability & Statistics Fundamentals Chapter 6). The standard rule of thumb: both $n \cdot p$ and $n \cdot (1-p)$ should be at least 5 in each group, or the approximation can break down for rare events.

Worked Example: A Conversion-Rate A/B Test

Control (A): $n_1=1000$, 80 conversions ($p_1=8\%$). Treatment (B): $n_2=1000$, 100 conversions ($p_2=10\%$). Validity check: $1000\times0.08=80$, $1000\times0.92=920$, both comfortably above 5 for both groups — the normal

approximation is safe to use.

QUANTITY	VALUE
p̂ (pooled)	180/2000 = 0.09
SE_pooled	≈ 0.01280
z	(0.10−0.08)/0.01280 ≈ 1.563
p-value (two-tailed)	≈ 0.118
Decision at α=0.05	0.118 > 0.05 → fail to reject H_0

The 95% confidence interval for the true difference, using the unpooled SE: `0.02 ± 1.96 × 0.01279 ≈ [-0.005, 0.045]` — a range spanning from a small *negative* effect to a fairly large positive one, consistent with the test's own failure to reach significance. The interval crossing zero and the non-significant p-value are two views of the exact same conclusion.

Sample Size Planning — Was This Test Big Enough?

Statistical power (`1 − β` , where `β` is Chapter 4's own Type II error rate) is the probability a test correctly detects a real effect of a given size, if one truly exists. Before running a test, sample size can be planned to hit a target power (conventionally 80%):

SAMPLE SIZE FORMULA (PER GROUP, TWO-PROPORTION TEST)

`n ≈ 2(z_α/2 + z_β)² × p̄(1−p̄) / (p₁−p₂)²` , with `z_α/2 = 1.96` (95% confidence) and `z_β ≈ 0.84` (80% power) as standard published constants

For the same 8%→10% effect actually being tested above (`p̄ = 0.09`):

THE TEST WAS UNDERPOWERED — THIS EXPLAINS THE NON-SIGNIFICANT RESULT

`n ≈ 2(1.96+0.84)² × 0.09(0.91) / (0.02)² ≈ 3,210` per group. The actual test only had `n = 1,000` per group — less than a third of what proper planning would have called for. This is almost certainly *why* the test failed to reach significance, even if the treatment genuinely does help: the sample was simply too small to reliably detect a real 2-percentage-point effect at this base rate, not proof that no real effect exists.

Why Smaller Effects Need Dramatically More Data

The effect size sits *squared* in the denominator of the sample size formula — a small change in the effect being detected has an outsized impact on the data required.

EFFECT TO DETECT	REQUIRED N PER GROUP (80% POWER, A=0.05)
8% → 10% (2 percentage points)	≈ 3,210
8% → 9% (1 percentage point)	≈ 12,195

Halving the effect size to detect very nearly **quadrupled** the required sample size — a direct consequence of that squared denominator, and a genuinely important planning reality: reliably detecting small improvements is expensive in data, often far more than intuition suggests.

A Real Trap: Peeking at Results Early

> **STOPPING AS SOON AS IT "LOOKS SIGNIFICANT" INFLATES THE FALSE-POSITIVE RATE**
>
> Checking a test's p-value repeatedly while it's still running, and stopping the moment it dips below 0.05, is a well-documented way to badly inflate the true false-positive rate above the intended α — related to, but distinct from, Chapter 4's own multiple-testing trap. A p-value naturally wanders up and down as data accumulates; checking it many times gives many chances for it to randomly dip below 0.05 at some point, even with zero real effect. The correct practice is to decide the sample size *before* the test starts (using the planning formula above) and only check the result once that predetermined size is reached.

A/B Testing in Code

```
import math

def standard_normal_cdf(z):
    return 0.5 * (1 + math.erf(z / math.sqrt(2)))

def two_proportion_z_test(x1, n1, x2, n2):
    p1, p2 = x1 / n1, x2 / n2
    p_pool = (x1 + x2) / (n1 + n2)
    se_pooled = math.sqrt(p_pool * (1 - p_pool) * (1/n1 + 1/n2))
    z = (p2 - p1) / se_pooled
    p_value = 2 * (1 - standard_normal_cdf(abs(z)))
    return z, p_value

z, p = two_proportion_z_test(x1=80, n1=1000, x2=100, n2=1000)
print(z, p)  # 1.563, 0.118 — matches the worked example

def required_sample_size(p1, p2, z_alpha2=1.96, z_beta=0.84):
    pbar = (p1 + p2) / 2
    return 2 * (z_alpha2 + z_beta)**2 * pbar * (1 - pbar) / (p2 - p1)**2

print(required_sample_size(0.08, 0.10))  # ~3210 per group
```

Hands-On Exercises

EXERCISE 1

Control: $n_1=500$, 60 conversions. Treatment: $n_2=500$, 85 conversions. First check the normal-approximation validity condition, then compute the pooled proportion, the z-statistic, and the p-value. State the decision at α=0.05.

View solution

EXERCISE 2

Using this chapter's own sample size formula, compute the required sample size per group (80% power, α=0.05) to detect a change from a 5% baseline conversion rate to a 6% conversion rate (a 1-percentage-point effect).

View solution

EXERCISE 3

A team plans a test for a fixed 4-week sample size, but decides to check the p-value every single day and stop the test as soon as it first drops below 0.05. Using this chapter's own peeking finding, explain why this practice produces a real false-positive rate higher than the intended 5%, even if the team genuinely stops testing the instant they see a "significant" result.

View solution

CHAPTER 6 QUICK REFERENCE

- **Two-proportion z-test:** pooled SE for the hypothesis test ($H_0: p_1=p_2$), unpooled SE for the confidence interval on the difference
- **Validity check:** $n \cdot p \geq 5$ and $n \cdot (1-p) \geq 5$ in each group before trusting the normal approximation
- **Statistical power** (1−β) is the probability of correctly detecting a real effect — plan sample size for it *before* running a test
- **Sample size formula:** $n \approx 2(z_{\alpha/2}+z_{\beta})^2\bar{p}(1-\bar{p})/(p_1-p_2)^2$ — effect size is squared in the denominator, so halving the effect roughly quadruples the required data
- A non-significant result on an underpowered test doesn't prove there's no effect — it may just mean there wasn't enough data to detect one
- **Never peek and stop early** — checking repeatedly and stopping at the first "significant" reading inflates the true false-positive rate well above α
- **Next chapter:** Correlation vs. causation

CHAPTER 7 OF 10

Correlation vs. Causation

Statistical Inference & Applied Statistics

Chapter 7 · Correlation vs. Causation

Every chapter so far tested a single variable, or compared two groups on one metric. This chapter asks a different question: do two *different* variables move together? And — the far more important half of the chapter — what an honest "yes" actually does and doesn't let you conclude.

The Pearson Correlation Coefficient

PEARSON'S R

`r = Σ((x−x̄)(y−ȳ)) / √(Σ(x−x̄)² × Σ(y−ȳ)²)` — ranges from `−1` (perfect negative linear relationship) through `0` (no linear relationship) to `+1` (perfect positive linear relationship)

Worked example: CPU usage (%) and response time (ms) across six servers:

CPU (%)	40	55	60	45	70	50
Response (ms)	120	150	165	130	190	140

`r ≈ 0.997` — an extremely strong positive linear relationship: as CPU usage rises, response time rises right along with it, almost perfectly linearly across this data.

The Question This Number Can't Answer

A correlation this strong is tempting to read as "high CPU usage *causes* slow responses." That might well be true here — but the number `r = 0.997` itself doesn't establish it. Any observed correlation between two variables has **four** possible explanations:

EXPLANATION	WHAT IT WOULD MEAN HERE
X causes Y	High CPU usage genuinely slows down response handling
Y causes X (reverse causation)	Slow-running requests themselves consume more CPU while they're stuck processing
A confounder Z causes both	A traffic surge drives both more CPU load *and* slower responses independently

EXPLANATION	WHAT IT WOULD MEAN HERE
Coincidence	With only 6 data points, a strong-looking correlation can arise from chance alone

The Classic Confounding Example

ICE CREAM SALES AND DROWNING DEATHS — GENUINELY, STRONGLY CORRELATED

Monthly ice cream sales and monthly drowning deaths correlate strongly and positively across an entire year. Ice cream doesn't cause drowning, and drowning certainly doesn't cause ice cream sales — both are driven by a real confounder: hot summer weather brings out both more swimmers and more ice cream buyers, independently. This is the textbook illustration of exactly why "X and Y move together" is not evidence that either one drives the other.

A more directly engineering-relevant version: "time spent on a page" correlates positively with "conversion rate" — but this is a strong candidate for **reverse causation**, not X causing Y. Users who are already leaning toward converting tend to read more carefully and spend longer on the page *because* they're interested — the extra time is a symptom of interest, not necessarily a cause of the sale.

What Actually Establishes Causation

THIS IS EXACTLY WHY CHAPTER 6'S A/B TEST MATTERS

Chapter 6's own randomized experiment sidesteps all four explanations above in one move: randomly assigning users to control or treatment means any potential confounder — traffic source, time of day, user type — gets distributed roughly equally between both groups *by construction*, not by hope. If the treatment group's conversion rate is still significantly different afterward, the *only* systematic difference between the groups was the treatment itself, ruling out both confounding and reverse causation. Pure observational correlation, however strong, can never make that same guarantee — only **randomization** can.

Correlation in Code

```python
def pearson_r(x, y):
    n = len(x)
    xbar = sum(x) / n
    ybar = sum(y) / n
    numerator = sum((x[i] - xbar) * (y[i] - ybar) for i in range(n))
    denom = (
        sum((xi - xbar) ** 2 for xi in x) *
        sum((yi - ybar) ** 2 for yi in y)
    ) ** 0.5
    return numerator / denom

cpu = [40, 55, 60, 45, 70, 50]
response = [120, 150, 165, 130, 190, 140]
print(pearson_r(cpu, response))  # 0.997...

# Python 3.10+ also has this built in:
import statistics
print(statistics.correlation(cpu, response))  # matches exactly
```

Hands-On Exercises

EXERCISE 1

A developer logs hours of sleep and number of bugs introduced the next day across six days: sleep = `[7, 5, 8, 4, 6, 7]`, bugs = `[1, 4, 0, 6, 3, 2]`. Compute the Pearson correlation coefficient, and state whether it's a strong positive, strong negative, or weak relationship.

View solution

EXERCISE 2

A team observes that teams with more Slack messages per day also close more tickets per day, with a strong positive correlation. Using this chapter's own four explanations, propose one plausible story for each of "X causes Y," "Y causes X," and "a confounder Z causes both" for this specific observation.

View solution

EXERCISE 3

Explain, using this chapter's own finding, exactly why randomly assigning users to control and treatment groups (Chapter 6's A/B testing) rules out confounding as an explanation for an observed difference, in a way that simply observing two already-correlated variables in the wild never can.

View solution

CHAPTER 7 QUICK REFERENCE

- **Pearson's r:** ranges −1 to +1, measuring the strength and direction of a *linear* relationship between two variables
- Any correlation has four possible explanations: X causes Y, Y causes X (reverse causation), a confounder Z causes both, or coincidence
- The classic example: ice cream sales and drowning deaths correlate strongly — both driven by hot weather, neither causing the other
- "Time on page" correlating with conversion is a strong candidate for reverse causation, not X causing Y
- Only **randomization** (Chapter 6's A/B test) rules out confounding by construction — pure observational correlation, however strong, cannot
- **Next chapter:** Linear regression as statistical inference

CHAPTER 8 OF 10

Linear Regression as Statistical Inference

Statistical Inference & Applied Statistics

Chapter 8 · Linear Regression as Statistical Inference

Linear Algebra Fundamentals' own Chapter 6 planted a forward reference: "fitting a straight line through a set of data points is solving a system of linear equations for the best-fit slope and intercept." This chapter is where that promise is finally kept — and it builds directly on Chapter 7's own correlation work, not separately from it.

Least Squares — Fitting the Best Line

Linear regression fits a line `y = mx + b` that best predicts `y` from `x`. "Best" means minimizing the sum of squared vertical distances between each actual point and the line — **least squares**.

LEAST SQUARES FORMULAS

`m = Σ((x−x̄)(y−ȳ)) / Σ((x−x̄)²)` `b = ȳ − m·x̄`

THE LINEAR ALGEBRA FUNDAMENTALS FORWARD REFERENCE, PAID OFF

That slope formula's numerator is *exactly* Chapter 7's own Pearson correlation numerator. These two least-squares formulas are, underneath, the solution to a small 2×2 system of linear equations — the "normal equations" — solved precisely the way Linear Algebra Fundamentals Chapter 6 described, using the exact same sum-of-products machinery Chapter 7 already introduced for correlation. Regression and correlation aren't two unrelated topics; regression is correlation's own numerator, repurposed to build a predictive line instead of just a strength score.

Worked Example: Fitting a Line to Chapter 7's Own Dataset

Reusing Chapter 7's exact CPU/response-time data:

QUANTITY	VALUE
Slope (m)	≈ 2.343
Intercept (b)	≈ 24.21
Fitted line	response ≈ 2.343 × CPU + 24.21

Residuals — What the Line Doesn't Capture

A **residual** is `actual − predicted` for each point — the vertical gap the line missed.

CPU	ACTUAL	PREDICTED	RESIDUAL
40	120	117.93	+2.07
55	150	153.07	−3.07
60	165	164.79	+0.21
45	130	129.64	+0.36
70	190	188.21	+1.79
50	140	141.36	−1.36

Small, mixed-sign residuals across the board — the line fits this data closely, with no obvious pattern left unexplained.

R^2 — The Square of Chapter 7's Own r

R-SQUARED

`R² = r²` — the proportion of variance in `y` "explained" by the linear relationship with `x`.

Chapter 7 already computed `r ≈ 0.997` for this exact dataset — no new calculation needed: `R² ≈ 0.997² ≈ 0.994`. About **99.4%** of the variation in response time across these six servers is explained by CPU usage alone in this linear model, leaving only about 0.6% unexplained.

Making a Prediction

Using the fitted line for a CPU value not in the original data, `CPU = 65%`: `response ≈ 2.343 × 65 + 24.21 ≈ 176.5ms`.

EXTRAPOLATION — A GENUINE DANGER

`CPU = 65%` sits comfortably inside the observed range (40–70%), so this prediction is reasonable. Plugging in `CPU = 100%` gives `≈ 258.5ms` — but nothing in the actual data says the relationship stays linear that far outside the observed range. A server pushed to 100% CPU might degrade far more sharply (or hit a hard ceiling) than a straight line extrapolated from 40–70% data could ever predict. Trust a fitted line only within the range of x-values it was actually built from.

CHAPTER 7'S OWN CAUTION STILL APPLIES IN FULL

An `R² = 0.994` is a genuinely excellent fit — and still says nothing about *causation*. High CPU usage causing slow responses remains only the most plausible of Chapter 7's own four explanations, not a proven one, regardless of how well the line fits.

Linear Regression in Code

```
def least_squares(x, y):
    n = len(x)
    xbar, ybar = sum(x)/n, sum(y)/n
    m = sum((x[i]-xbar)*(y[i]-ybar) for i in range(n)) / sum((xi-xbar)**2 for xi in x)
    b = ybar - m * xbar
    return m, b

cpu = [40, 55, 60, 45, 70, 50]
response = [120, 150, 165, 130, 190, 140]
m, b = least_squares(cpu, response)
print(m, b)  # 2.343, 24.21

def predict(x_new, m, b):
    return m * x_new + b

print(predict(65, m, b))  # 176.5
```

Hands-On Exercises

EXERCISE 1

Reusing Chapter 7's own sleep/bugs dataset (`sleep = [7,5,8,4,6,7]`, `bugs = [1,4,0,6,3,2]`), fit a least-squares regression line, and use it to predict the number of bugs for `6.5` hours of sleep.

View solution

EXERCISE 2

Chapter 7 computed `r ≈ −0.985` for the sleep/bugs dataset. Using this chapter's own `R² = r²` relationship, compute R^2 and state, in plain terms, what percentage of the variation in bugs introduced is explained by hours of sleep in this model.

View solution

EXERCISE 3

Using this chapter's own main worked example (CPU vs. response time, data ranging from 40% to 70% CPU), explain specifically why predicting response time at `CPU = 5%` is just as risky as the chapter's own `CPU = 100%` extrapolation example, even though 5% is a much smaller, seemingly more "reasonable" number.

View solution

CHAPTER 8 QUICK REFERENCE

- **Least squares:** `m = Σ((x−x̄)(y−ȳ))/Σ((x−x̄)²)`, `b = ȳ − m·x̄` — the same numerator as Chapter 7's Pearson r, repurposed to build a predictive line
- This is literally Linear Algebra Fundamentals Chapter 6's own forward reference: fitting a line is solving a small system of linear equations (the normal equations)
- **Residual** = actual − predicted, for each point
- **$R^2 = r^2$** — the proportion of variance in y explained by the linear relationship with x
- **Extrapolation risk:** a fitted line is only trustworthy within the range of x-values actually observed
- A high R^2 is still not proof of causation — Chapter 7's own correlation-vs-causation caution applies in full
- **Next chapter:** Bayesian inference and updating beliefs

CHAPTER 9 OF 10

Bayesian Inference & Updating Beliefs

Statistical Inference & Applied Statistics

Chapter 9 · Bayesian Inference & Updating Beliefs

Chapter 1 promised this chapter would generalize Probability & Statistics Fundamentals' own Bayes' Theorem — applied there to a single, fixed piece of evidence — into an ongoing process. Here's the whole idea in one sentence: **yesterday's posterior is today's prior.**

Two Genuinely Different Philosophies

Every chapter of this course except this one has quietly worked in the **frequentist** framework: a population parameter (a true mean, a true conversion rate) is a fixed, unknown constant, and probability describes the long-run behavior of the *estimation method* across repeated sampling — precisely Chapter 3's own correct interpretation of a confidence interval. The **Bayesian** framework treats the parameter itself as uncertain, with its own probability distribution representing genuine belief — a distribution that gets updated, piece by piece, as evidence arrives.

> **CHAPTER 3'S OWN WARNING, FINALLY RESOLVED**
>
> Chapter 3 warned that a 95% confidence interval does *not* mean "95% probability the true value is in this interval" — that intuitive-sounding claim is actually the correct definition of a **Bayesian credible interval** instead, a genuinely different object built from a posterior distribution rather than repeated sampling. The interpretation everyone instinctively reaches for isn't wrong about statistics in general — it's just describing the *other* framework.

Building an actual numeric credible interval needs machinery beyond this course's own scope (conjugate priors, the Beta distribution for proportions) — genuinely out of scope here, per Chapter 1's own boundary. This chapter covers Bayesian *reasoning*, not Bayesian computation at that level.

Sequential Updating: Yesterday's Posterior, Today's Prior

Reusing Probability & Statistics Fundamentals Chapter 4's own spam-filter numbers exactly: `P(spam) = 0.40`, `P("free"|spam) = 0.30`, `P("free"|not spam) = 0.05`. That chapter computed the posterior once: `P(spam|"free") = 0.80`.

Now a second, independent signal arrives on the same email — a suspicious link, with `P(link|spam) = 0.60`, `P(link|not spam) = 0.02`. The trick: **the 0.80 posterior from round one becomes the prior for round two**.

ROUND	PRIOR GOING IN	EVIDENCE	POSTERIOR
1	0.40	Contains "free"	0.80
2	0.80 (round 1's posterior)	Contains a suspicious link	≈ 0.992

Two pieces of corroborating evidence, applied one at a time, drove belief from a 40% baseline all the way to over 99% — each round using nothing more than Fundamentals Chapter 4's own Bayes' Theorem, applied again with an updated starting point.

THIS IS EXACTLY HOW A NAIVE BAYES SPAM CLASSIFIER WORKS

Real spam filters combine many word-level and metadata signals this same way — updating a probability sequentially (or equivalently, all at once assuming the evidence pieces are conditionally independent given the hypothesis, the "naive" assumption naive Bayes is named for). Data Science & ML's own `ml1` covers this as an actual classification algorithm; here it's the same underlying math, without the ML framing.

A/B Testing Results, Read the Bayesian Way

Some modern experimentation platforms report results as "94% probability B beats A" rather than Chapter 6's own p-value framing. That's a genuinely Bayesian statement — a direct probability about which variant is actually better — distinct from, and not directly interchangeable with, a frequentist p-value. Both are legitimate, real tools in active use; knowing which framework a specific number came from is what makes it possible to interpret it correctly.

The Honest Tradeoff: Where the Prior Comes From

A BAYESIAN ANALYSIS DEPENDS ON A CHOICE, NOT JUST THE DATA

Every Bayesian update starts from a prior — and that prior has to come from somewhere: past data, domain expertise, or a deliberately "uninformative" default. Two reasonable analysts can pick two different reasonable priors and land on genuinely different posteriors from the identical evidence. This is a real, honest tradeoff, not a flaw to be embarrassed about — frequentist methods avoid needing a prior at all, at the cost of the less intuitive interpretation Chapter 3 already covered. Neither framework is simply "better"; they trade one kind of honesty for another.

Sequential Bayesian Updating in Code

```python
def bayes_update(prior, p_evidence_given_h, p_evidence_given_not_h):
    p_not_h = 1 - prior
    p_evidence = (p_evidence_given_h * prior) + (p_evidence_given_not_h * p_not_h)
    return (p_evidence_given_h * prior) / p_evidence

# Round 1: reusing Fundamentals Ch.4's own spam filter numbers
posterior_1 = bayes_update(prior=0.40, p_evidence_given_h=0.30, p_evidence_given_not_h=0.05)
print(posterior_1)  # 0.8

# Round 2: the round-1 posterior becomes the round-2 prior
posterior_2 = bayes_update(prior=posterior_1, p_evidence_given_h=0.60, p_evidence_given_not_h=0.02)
print(posterior_2)  # 0.9917...
```

Hands-On Exercises

EXERCISE 1

Reusing Probability & Statistics Fundamentals Chapter 4's own fraud-detection example (`P(fraud) = 0.001` , `P(flagged|fraud) = 0.95` , `P(flagged|not fraud) = 0.02` , giving a posterior of `P(fraud|flagged) ≈ 0.0454`), a second signal now fires on the same transaction: an unusual login location, with `P(location|fraud) = 0.70` , `P(location|not fraud) = 0.10` . Using this chapter's own sequential-updating method, compute the new posterior after both pieces of evidence.

View solution

EXERCISE 2

A colleague says "a 90% confidence interval and a 90% credible interval mean the same thing, just from two different calculation methods." Using this chapter's own frequentist-vs-Bayesian distinction, explain specifically what's wrong with this claim.

View solution

EXERCISE 3

Two analysts investigate the same new fraud-detection signal. Analyst A, who has seen many false alarms from similar signals before, starts with a skeptical prior. Analyst B, newer to the team, starts with a more neutral prior. Given the exact same evidence, explain why they could reasonably reach different posteriors, and why this doesn't mean one of them made a mathematical error.

View solution

CHAPTER 9 QUICK REFERENCE

- **Frequentist:** parameters are fixed constants; probability describes the estimation method's long-run behavior (Chapter 3's own CI interpretation)
- **Bayesian:** parameters have their own probability distribution, representing genuine belief, updated as evidence arrives
- A 95% *credible* interval genuinely does mean "95% probability the true value is here" — the intuitive interpretation Chapter 3 warned doesn't apply to a frequentist CI
- **Sequential updating:** yesterday's posterior becomes today's prior — repeated application of Fundamentals Chapter 4's own Bayes' Theorem
- This same mechanism, applied to many signals, is the basis of a naive Bayes classifier — real ML tooling, same underlying math
- A Bayesian analysis depends on the prior chosen — a real, honest tradeoff against frequentist methods' own less intuitive interpretation, not a flaw
- **Next chapter:** Capstone — designing and analyzing a real experiment

CHAPTER 10 OF 10

Capstone — Designing and Analyzing a Real Experiment

Statistical Inference & Applied Statistics

Chapter 10 · Capstone — Designing and Analyzing a Real Experiment

One continuous worked experiment, touching every chapter of this course in the order a real analyst would actually reach for each idea: testing whether a redesigned checkout button genuinely improves conversion, from a historical baseline through a properly powered experiment to a closing Bayesian reframing.

A Full Worked Experiment — Testing a Redesigned Checkout Button

1 — THE HISTORICAL BASELINE IS A SAMPLE, NOT THE TRUTH (CH.2)

Before designing anything, the team pulls 400 historical sessions: 32 conversions, `p = 8%`. This is a **sample** of past traffic, not a guaranteed future rate — it carries genuine sampling error, and past traffic mix may not perfectly match what the actual test window will see. This number is a starting estimate, not a fact to design blindly around.

2 — QUANTIFYING HOW MUCH TO TRUST THAT BASELINE (CH.3)

Treating 8% as a proportion estimate: `SE = √(0.08×0.92/400) ≈ 0.0136`. The 95% confidence interval: `0.08 ± 1.96×0.0136 ≈ [5.3%, 10.7%]`. The true baseline conversion rate could honestly be anywhere in that fairly wide range — a useful, humbling number to have before committing to a test design built on the single point estimate of 8%.

3 — STATING THE FORMAL HYPOTHESES (CH.4)

`H₀: p_control = p_treatment` (the button color makes no real difference) against `H₁: p_control ≠ p_treatment`, at the standard `α = 0.05`. Nothing about the experiment's own numbers is examined yet — the rules are fixed first, deliberately, to avoid the peeking trap Chapter 6 warned about.

4 — AN A/A SANITY CHECK BEFORE THE REAL TEST (CH.5)

Before trusting the traffic-splitting mechanism itself, the team runs an **A/A test** — two groups, both shown the *old* button, compared on average session duration to confirm the split itself introduces no artificial bias. Group 1: `n=30`, mean 145s, `s=40s`. Group 2: `n=32`, mean 138s, `s=38s`. Welch's t-test: `SE_diff ≈ 9.92`, `t ≈ 0.705`, `df = 29`, critical value ≈ 2.045. `|0.705| < 2.045` → fail to reject, exactly the desired outcome — the split mechanism itself shows no detectable bias, so the actual A/B test can proceed with confidence in the infrastructure.

5 — RUNNING THE PROPERLY POWERED TEST (CH.6)

Using Step 1's own 8% baseline and wanting to detect a lift to 10% (Chapter 6's own worked effect size), proper sample-size planning calls for **≈3,210 per group** — not the smaller, underpowered pilot Chapter 6's own example ran with. This time, the team runs the test to that full planned size: control `n=3,210`, 257 conversions (≈8.01%); treatment `n=3,210`, 321 conversions (10.00%).

QUANTITY	VALUE
Pooled p̂	≈ 0.0900
SE_pooled	≈ 0.00714
z	≈ 2.791
p-value	≈ 0.0053

PROPER PLANNING FINDS WHAT THE PILOT MISSED

`0.0053 < 0.05` — this time, **reject H_0**. The exact same 8%→10% effect that Chapter 6's own underpowered `n=1,000` pilot failed to detect (`p≈0.118`) becomes clearly significant once the sample size is actually planned properly. Nothing about the real effect changed — only whether there was enough data to reliably see it.

6 — RESISTING A TEMPTING OBSERVATIONAL SHORTCUT (CH.7)

Digging into the treatment group's data, engaged sessions (more product pages viewed) also tend to run longer — `r ≈ 0.998` between pages viewed and session duration across a sample. It's tempting to conclude "showing users more product pages makes them stay longer," but per Chapter 7's own caution, this observational correlation could just as easily be reverse causation: genuinely interested users naturally view more pages *and* stay longer, with neither driving the other. Only Step 5's own randomized comparison — not this correlation — can support a causal claim.

7 — A REGRESSION-BASED FOLLOW-UP (CH.8)

Fitting a least-squares line to the same pages-viewed/duration data: `duration ≈ 40.77 × pages + (−24.62)`, with `R² ≈ 0.996` — a striking fit. Predicting duration for a session viewing 7 pages: `≈ 260.8s`. Since the observed data only ranged up to 6 pages viewed, this prediction is already a mild extrapolation, per Chapter 8's own caution — worth flagging honestly rather than reporting with false confidence.

8 — A CLOSING BAYESIAN REFRAMING (CH.9)

Step 5's result is a frequentist one: `H₀` rejected at `α=0.05`, a binary decision. A Bayesian analyst, starting from a skeptical prior (most small UI tweaks historically move conversion very little), would update that prior using this experiment's own strong evidence (`p≈0.0053`, a properly powered result) and land on a substantially higher posterior belief the button change genuinely helps — a graded probability statement rather than a reject/fail-to-reject binary. Per Chapter 9's own honest scope boundary, computing that exact posterior number needs machinery (a specified prior distribution, conjugate updating) beyond this course — but the qualitative direction is clear either way: strong, properly powered evidence moves a skeptical starting belief substantially, regardless of which framework reports the final number.

This is, in essence, exactly what a rigorous, real A/B test looks like end to end — every step traceable to a specific chapter of this course, none of it a shortcut around the planning and honesty each one demanded.

What This Course Doesn't Cover

In the interest of an honest accounting: **advanced experimental design** (multi-armed bandits, sequential testing), **machine learning model evaluation**, and **full computational Bayesian methods** were all named in Chapter 1 as deliberately out of scope. This course built the core inferential toolkit those more specialized areas each build on, not a substitute for them.

This Course's Throughline, Restated

REASONING HONESTLY UNDER INCOMPLETE INFORMATION

Every chapter in this course answered a version of the same question: how much can a limited, incomplete sample actually tell you about the truth, and how do you act correctly under the uncertainty that remains? The capstone above used exactly eight ideas — sampling, confidence intervals, hypothesis testing, the t-test, proper A/B test planning, correlation's limits, regression, and Bayesian updating — across one continuous, realistic experiment. That's the real payoff: not a single flashy technique, but a disciplined process that catches its own mistakes (an underpowered pilot, a tempting correlational shortcut) before they become false conclusions.

Where This Course Connects

Together with its sibling course, this completes the Probability & Statistics arc within Maths for Programmers — Probability & Statistics Fundamentals built the forward-reasoning vocabulary and distributions; this course built the backward-reasoning discipline for drawing honest conclusions from real, limited data. Both connect directly to Technical Support's own diagnostic material (`perfdiag1`/`incident1`) and to Data Science & ML's own `ds1`/`ml1`, where this exact inferential toolkit gets applied at production scale.

Hands-On Exercises

EXERCISE 1

A different historical baseline shows 45 conversions out of 500 sessions. Using this chapter's own Steps 1–2 technique, compute the point estimate and its 95% confidence interval.

View solution

EXERCISE 2

A team runs an A/A test comparing two groups both shown the same design: Group 1 `n=25`, mean=200, `s=30`; Group 2 `n=28`, mean=215, `s=35`. Using this chapter's own Step 4 technique, compute Welch's t-statistic and the conservative degrees of freedom. Using the critical value 2.064 (df=24), state whether this A/A test passes its own sanity check.

View solution

EXERCISE 3

For each of the eight steps in this chapter's own worked experiment, name the specific inferential-statistics topic it relied on, without looking back at the step labels — just from the description of what each step actually does.

View solution

CHAPTER 10 QUICK REFERENCE

- **Full worked experiment:** a sampled baseline (Ch.2) → a confidence interval on it (Ch.3) → formal hypotheses (Ch.4) → an A/A sanity check via t-test (Ch.5) → a properly powered A/B test (Ch.6) → resisting an observational shortcut (Ch.7) → a regression follow-up (Ch.8) → a Bayesian reframing (Ch.9)
- Out of scope: advanced experimental design, ML model evaluation, and full computational Bayesian methods — each reserved for its own specialized study

- This course's throughline: a disciplined process for reasoning honestly under incomplete information, catching its own mistakes (underpowered pilots, tempting correlations) before they become false conclusions
- This course, together with Probability & Statistics Fundamentals, is the direct foundation under Technical Support's diagnostic material and Data Science & ML's own applied work
- **Course complete** — Statistical Inference & Applied Statistics, 10 chapters, from sampling to Bayesian updating