10 *Software and Tools*

Every method in this book was developed by hand: compute a statistic, look up a table, state a decision. That discipline is deliberate, because a person who cannot run a test by hand cannot judge whether a computer ran it correctly. This chapter now hands the arithmetic to the computer. It surveys the statistical software an industrial engineer will actually meet—Minitab, Excel, Python, and R—and then walks through the two tools used in this course, Minitab and Python, showing how to run every method of Chapters 2 through 9: confidence intervals, one-sample and two-sample tests, regression, and the analysis of variance. For each method the chapter shows what you type or click, what the output looks like, and—most importantly—how each field of that output maps back to an equation in this book. The chapter closes with two practical topics: why software p-values differ slightly from table lookups, and how to use AI assistants for statistical work without being misled by them. The division of labor never changes: software computes; you formulate, check assumptions, and interpret.

Pre-class quiz. *Try these before lecture; the answers follow at the end of the quiz.*

Q1. A hand calculation using the t table bounds a p-value as $0.02 < p < 0.05$. Software reports $p = 0.0297$ for the same data. Do the two approaches ever lead to different decisions at $\alpha = 0.05$?

Q2. Which of the following does statistical software *not* do for you: (A) compute the test statistic; (B) find the exact p-value; (C) decide whether a t-test is the right test for your data; (D) compute the confidence interval bounds.

Q3. Software reports a two-sided p-value of 0.08 for a one-sample t-test. Your alternative is one-sided, $H_1 : \mu > \mu_0$, and the sample mean came out *above* μ_0. What is the one-sided p-value?

Answers.

Q1. *No, not here.* The table bound and the exact value agree that $p < 0.05$, so both reject H_0. The software value is more precise, not more correct: both are computed from the same statistic and the same distribution. A conflict could only appear when the table bound straddles α (for example $0.02 < p < 0.05$ with $\alpha = 0.025$), and then the exact value settles it.

Q2. (C). Software executes whatever test you select, on whatever data you give it, whether or not the test is appropriate. Choosing the method, checking its assumptions, and interpreting the result remain your job.

Q3. $0.08/2 = 0.04$. When the observed effect points in the direction of H_1, the one-sided p-value is half the two-sided one, because only one tail of the reference distribution counts. Please pay attention to the direction: if the sample mean had come out *below* μ_0, the one-sided p-value would be $1 - 0.08/2 = 0.96$, not 0.04.

Lecture 16 starts here — *Software tools*

Lecture 16. This section is covered by Lecture 16.

Reference. Montgomery & Runger, 6th ed.; computer output boxes appear throughout Chapters 8–14.

Statistical software. A program that computes statistics, p-values, interval bounds, and diagnostic plots from data you supply, using the same formulas as this book.

10.1 From Tables to Tools

What we have been doing by hand has three steps: compute a test statistic (Z_0, T_0, χ^2_0, or F_0), look up a critical value in an appendix table, and bound the p-value between two table entries. Software replaces the second and third steps and speeds up the first. Specifically, software gives us:

- **Exact p-values.** Instead of the bound $0.02 < p < 0.05$ read from a table, software reports $p = 0.0297$.
- **Graphical diagnostics.** Residual plots, normal probability plots, and box plots are produced automatically, so checking assumptions becomes routine rather than laborious.
- **Scale.** A regression with six predictors and $n = 500$ observations requires the matrix computation $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ of (7.4), which is impractical by hand and instantaneous by machine.

Exact p-value. The p-value computed from the continuous reference distribution itself, rather than bounded between two printed table entries.

Software speeds up *calculation*. It does not replace understanding. Every output field in this chapter is a number you already know how to compute; the software merely computes it faster and to more decimal places.

Four tools dominate practice, and Table 10.1 compares them. In ISE 315 we focus on understanding the methods; any of these tools can execute them. This chapter works with two: **Minitab**, the menu-driven package that is the de facto standard in industrial engineering and quality work, and **Python** with the `scipy.stats` module, which is free, programmable, and reproducible.

Tool	Strengths	Limitations	Typical use
Minitab	Menu-driven; built for quality and industrial statistics	Commercial license; less flexible	Six Sigma, quality control, design of experiments in industry
Excel	Widely available; familiar interface	Limited statistical functions; rounding issues	Quick calculations, data entry
Python (`scipy`)	Free; fully programmable; reproducible	Requires coding knowledge	Research, automation, machine learning
R	Purpose-built for statistics; rich packages	Steeper learning curve	Academic research, biostatistics

Table 10.1. The statistical software landscape. This course uses Minitab and Python.

Startup lens. An early-stage startup rarely owns a Minitab license, but `scipy` costs nothing. Every test in this chapter can be run in a free Python notebook, which also becomes a permanent, re-runnable record of the analysis behind each product decision.

Reference. Lecture 16; see also Montgomery & Runger, 6th ed., Section 9-1 on the structure of a test.

10.2 *The Division of Labor*

Before touching any menu or writing any code, fix in your mind what the software will and will not do. Table 10.2 draws the line.

Software handles this	You still must do this
Compute test statistics (Z_0, T_0, χ^2_0, F_0)	Choose the correct test for the problem
Look up exact p-values	Interpret the p-value in context
Calculate CI bounds	Select the right confidence level
Degrees of freedom (including Welch's (5.7))	Verify assumptions (normality, independence)
Residual plots	Judge whether the assumptions are met
Regression coefficients	Decide which variables to include
ANOVA decomposition	Explain what the results mean to a stakeholder

Table 10.2. The division of labor between the engineer and the software.

Please pay attention here, because this is the trap that produces most software-era statistical errors: **software will happily run a t-test on non-normal data, or fit a regression with meaningless predictors. It does not check whether your analysis makes sense.** A t-test on highly skewed data with $n = 5$ still produces a p-value; that p-value may simply not be valid. The four tasks that remain yours are exactly the ones this book has trained:

1. **Problem formulation.** Writing H_0 and H_1 correctly (4.1), one-sided versus two-sided, and choosing α from the engineering consequences of each error type (4.2).

2. **Test selection.** σ known $\to z$; σ unknown $\to t$; variance $\to \chi^2$; two variances $\to F$; paired versus independent samples (Procedure 5.2). The software does not decide any of this for you.

3. **Assumption checking.** Is the population approximately normal? Is n large enough for the CLT (2.8)? Are the samples truly independent? For proportions, are $n\hat{p} \geq 5$ and $n(1-\hat{p}) \geq 5$?

4. **Interpretation and communication.** Translating ''reject H_0 at $\alpha = 0.05$'' into an engineering decision, and reporting the practical size of the effect, not only its statistical significance.

Engineering judgment. Item 2 is engineering knowledge in disguise. Recognizing that before/after measurements on the same fifteen machines are *paired* data comes from knowing how the data were collected, not from anything visible in the spreadsheet. No menu can recover a study design you did not notice.

Figure 10.1 shows the resulting workflow. You formulate the problem and choose the settings; the software computes; you interpret and communicate. The arrows go both ways because the diagnostics the software returns (residual plots, normality checks) feed back into your judgment about whether the analysis is valid.

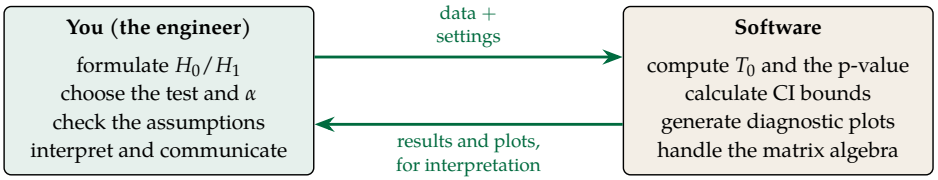

Figure 10.1. The statistical software workflow. Judgment flows in; arithmetic flows back out.

Insight. Everything the software prints is a quantity this book has already derived. A "T-Value" is (4.6); an "SE Mean" is $s/\sqrt{n}$ from (2.5); an "F-Value" in a regression output is (6.20). Reading software output is therefore not a new skill. It is a translation exercise: match each printed field to the equation it implements. The tables in this chapter are that dictionary.

Reference. Montgomery & Runger, 6th ed., Sections 8-1 to 8-4.

Menu path. The sequence of menu selections that reaches a command in a menu-driven package, written like `Stat` → `Basic Statistics` → `1-Sample t`.

Session window. The text pane where Minitab prints its numerical output; the printed report you will be asked to read on exams and in practice.

10.3 *Confidence Intervals in Software*

Minitab organizes all one-sample methods under `Stat` → `Basic Statistics`. For each interval you provide either a column of raw data or the summary statistics (n, $\bar{x}$, and s or σ), plus a confidence level. Consider first the z interval of (3.1), which requires a known σ. For a pipeline wall-thickness study with $n = 25$, $\bar{x} = 0.2731$ in, and known $\sigma = 0.02$ in, Minitab's `1-Sample Z` returns the following session-window output.

```
Minitab --- 1-Sample Z: WallThickness

 N      Mean     StDev   SE Mean   95% CI for μ
25   0.27310   0.02000   0.00400   (0.26527, 0.28093)

μ: population mean of WallThickness
Known standard deviation = 0.02
```

Verify it by hand against (3.1): the field `SE Mean` is $\sigma/\sqrt{n} = 0.02/\sqrt{25} = 0.004$, and

$$\bar{x} \pm z_{0.025}\,\frac{\sigma}{\sqrt{n}} = 0.2731 \pm 1.96 \times 0.004 = (0.2653,\ 0.2809),$$

which matches the printed interval to rounding. This hand check is not busywork. It is how you confirm that the software ran the interval you intended—here, a z interval with σ declared known.

When σ is unknown, the correct menu is `1-Sample t`, which implements the t interval of (3.5). For $n = 25$ monthly energy-cost measurements:

```
Minitab --- 1-Sample t: EnergyCost

 N     Mean    StDev   SE Mean   95% CI for μ
25   330.60   154.20     30.84   (266.94, 394.26)

μ: population mean of EnergyCost
```

Note the three differences from the z output: the software used the sample standard deviation $s = 154.20$ instead of a known σ; it multiplied by $t_{0.025,24} = 2.064$ instead of $z_{0.025} = 1.96$; and the interval is wider than a z interval on the same data would be, reflecting the extra uncertainty from estimating σ. Nothing in the output says "I used a t multiplier"—you know it because you chose the `1-Sample t` menu, and you can confirm it because $2.064 \times 30.84 = 63.65$ matches the half-width $(394.26 - 266.94)/2 = 63.66$.

In Python, the same interval is a few lines of `scipy.stats`. The code computes the pieces of (3.5) explicitly, which makes the correspondence to the book exact:

```
from scipy import stats
import numpy as np


n = len(data)
xbar = np.mean(data)
s    = np.std(data, ddof=1)   # ddof=1 gives the sample SD
t_crit = stats.t.ppf(0.975, df=n-1)  # the exact t table lookup
half = t_crit * s / np.sqrt(n)
ci = (xbar - half, xbar + half)
```

The function `stats.t.ppf(0.975, df)` is the software version of the t table: it returns the quantile $t_{0.025,\,\mathrm{df}}$ exactly, where the appendix table rounds to three decimals. The companion functions `stats.norm.ppf` and `stats.chi2.ppf` replace the z and χ^2 tables in the same way, so the intervals (3.6) for σ^2 and (3.7) for p follow the same pattern. Table 10.3 collects the full map for Chapter 3.

Interval	Minitab menu	Python	Implements
CI for μ, σ known	`Stat` → `Basic Statistics` → `1-Sample Z`	`norm.ppf` + formula	(3.1)
CI for μ, σ unknown	`1-Sample t`	`t.ppf` + formula, or `t.interval`	(3.5)
CI for σ^2	`1 Variance`	`chi2.ppf` + formula	(3.6)
CI for p	`1 Proportion`	`norm.ppf` + formula	(3.7)

Table 10.3. Method-to-tool map for the confidence intervals of Chapter 3. All Minitab paths start from `Stat` → `Basic Statistics`.

Procedure 10.1 — Running and checking a confidence interval in software.

1. Decide which interval the problem needs (Procedure 3.5): is σ known, is the target a mean, a variance, or a proportion? The software will not decide this.
2. Supply the data column or the summary statistics (n, $\bar{x}$, s or σ; for proportions, the event count and the number of trials), and set the confidence level.
3. Run the matching command from Table 10.3 and read the interval from the output.
4. Check one anchor value by hand: confirm that `SE Mean` equals $s/\sqrt{n}$ (or $\sigma/\sqrt{n}$), and that the half-width equals the multiplier times the standard error.
5. State the interval in context, with units, exactly as in Chapter 3.

Example 10.1 — Guardrail latency: reading a t-interval output

An AI safety team measures the added response latency (ms) that a new guardrail filter imposes on a language model. They collect $n = 16$ measurements and run Minitab's `1-Sample t`, also asking for a test of H_0: $\mu = 200$

against $H_1: \mu \neq 200$ at $\alpha = 0.05$:

```
Minitab --- 1-Sample t: Latency

 N     Mean   StDev   SE Mean   95% CI for μ
16   212.50   24.80      6.20   (199.29, 225.71)

Null hypothesis        H0: μ = 200
Alternative hypothesis H1: μ ≠ 200

T-Value   P-Value
   2.02     0.062
```

Verify the interval and the T-Value by hand, state the test decision, and explain how the interval and the test agree.

Solution.

Step 1 — Verify the standard error and the interval.

By (2.5), $s/\sqrt{n} = 24.80/\sqrt{16} = 6.200$, matching `SE Mean`. With $t_{0.025,15} = 2.131$, the interval (3.5) is

$$212.50 \pm 2.131 \times 6.200 = 212.50 \pm 13.21 = (199.29,\ 225.71),$$

matching the output.

Step 2 — Verify the test statistic.

By (4.6), $T_0 = (212.50 - 200)/6.200 = 12.50/6.200 = 2.016$, printed as 2.02.

Step 3 — Decide.

The exact p-value is $0.062 > 0.05$, so we fail to reject H_0 at $\alpha = 0.05$: the data do not establish that the mean added latency differs from 200 ms.

Step 4 — The CI–HT connection.

The hypothesized value 200 lies *inside* the 95 % interval (199.29, 225.71)—barely. A two-sided test at $\alpha = 0.05$ rejects exactly when the hypothesized value falls outside the 95 % CI, so the interval and the p-value tell one consistent story. The answer is $\boxed{\text{fail to reject } H_0;\ 95\%\ \text{CI } (199.29,\ 225.71) \text{ contains } 200}$.

Safety lens. A near-boundary result like $p = 0.062$ is exactly where a safety report must show the full output, not just the verdict. The interval says the mean latency could plausibly be as high as 226 ms; whether that is acceptable is a system-requirement question the p-value alone cannot answer.

Reference. Montgomery & Runger, 6th ed., Sections 9-1 to 9-5.

10.4 Hypothesis Tests in Software

The same `Basic Statistics` menus that produce intervals also run the one-sample tests of Chapter 4: in each dialog you tick ''Perform hypothesis test,'' enter the hypothesized value, and choose the direction of H_1. The output then adds a test block to the descriptive block. For a chemical-yield study with $n = 10$, $\bar{x} = 82.5$, $s = 3.2$, testing $H_0\colon \mu = 80$ at $\alpha = 0.05$:

```
Minitab --- 1-Sample t: Yield

 N     Mean   StDev   SE Mean   95% CI for μ
10   82.500   3.200     1.012   (80.212, 84.788)

Null hypothesis        H0: μ = 80
Alternative hypothesis H1: μ ≠ 80

T-Value   P-Value
   2.47     0.035
```

By hand, $T_0 = (82.5 - 80)/(3.2/\sqrt{10}) = 2.5/1.0119 = 2.471$ from (4.6), and the t table with 9 degrees of freedom only brackets the p-value: $0.02 < p < 0.05$. The software reports the exact value, $p = 0.035$, and the decision is **reject** H_0. Notice also the CI–HT connection in the same output: $80 < 80.212$, so the hypothesized mean lies *outside* the 95 % CI—consistent with rejection at $\alpha = 0.05$.

The variance test of (4.7) lives under `1 Variance`. For pipeline wall thickness with $n = 20$ and $s = 0.018$ in, testing $H_0\colon \sigma^2 = 0.0004$:

```
Minitab --- 1 Variance: WallThickness

Null hypothesis        H0: σ² = 0.0004
Alternative hypothesis H1: σ² ≠ 0.0004
Method: Chi-Square

Statistic   DF   P-Value
    15.39   19     0.697

95% CI for σ²: (0.000162, 0.000717)
```

Check the statistic against (4.7): $\chi^2_0 = (20-1)(0.018)^2/0.0004 = 19 \times 0.000324/0.0004 = 15.39$. The χ^2 table only says $p > 0.10$; the software says $p = 0.697$, and we fail to reject H_0. Minitab also reports the CI for σ^2 from (3.6) and, for non-normal data, an alternative called the Bonett method—a reminder that the χ^2 procedure is sensitive to non-normality.

One default deserves special care. **Most software reports a two-sided p-value unless told otherwise.** If your alternative is one-sided, either select the correct direction in the dialog, or convert the reported two-sided value p_{two} yourself:

$$p_{\text{one}} = \begin{cases} \dfrac{p_{\text{two}}}{2}, & \text{if the sample effect points in the direction of } H_1, \\ 1 - \dfrac{p_{\text{two}}}{2}, & \text{if it points the other way.} \end{cases} \tag{10.1}$$

The second case matters. Halving a two-sided p-value when the observed mean sits on the wrong side of μ_0 manufactures significance out of evidence that actually points *against* the alternative. Exercise 10.9 shows this error in the wild.

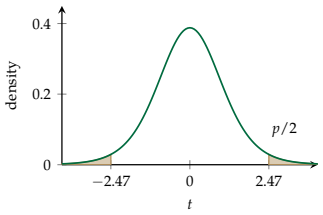

Figure 10.2. The exact two-sided p-value for the yield test: the two shaded tails of the t_9 density beyond ± 2.47 sum to 0.035. The table can only bracket this area between printed columns.

Python mirrors the same tests through `scipy.stats`. Table 10.4 lists the key functions; the workhorse pattern is that `ttest_1samp`, `ttest_ind`, and `ttest_rel` return the pair $(T_0,$ two-sided $p)$ directly, while the distribution objects `norm`, `t`, `chi2`, and `f` supply `.cdf()` for p-values and `.ppf()` for critical values whenever you compute a statistic yourself.

Our test	Python function	Returns
One-sample z (CI/test)	`norm.cdf()`, `norm.ppf()`	CDF values, critical values
One-sample t-test	`ttest_1samp(data, mu0)`	T_0, p-value
Two-sample t-test	`ttest_ind(data1, data2)`	T_0, p-value
Paired t-test	`ttest_rel(before, after)`	T_0, p-value
χ^2 test on variance	`chi2.cdf()`, `chi2.ppf()`	CDF values, critical values
F-test on variances	`f.cdf()`, `f.ppf()`	CDF values, critical values
Proportion z-test	`proportions_ztest()`	Z_0, p-value

Table 10.4. Key functions in `scipy.stats` for the tests of Chapters 4 and 5. (`proportions_ztest` is in the companion package `statsmodels`.)

A complete one-sample t-test in Python, matching the yield example above:

```
from scipy import stats
import numpy as np

data = [82.1, 84.3, 79.5, 83.8, 81.2, 85.1, 80.9, 83.4, 82.7, 81.5]

# H0: mu = 80 vs H1: mu != 80, alpha = 0.05
mu_0 = 80
t_stat, p_value = stats.ttest_1samp(data, mu_0)
```

```
print(f"T-statistic: {t_stat:.4f}")
print(f"P-value:     {p_value:.4f}")
```

Script. A saved sequence of commands that can be re-run on new data. Scripts make an analysis reproducible and auditable in a way that mouse clicks are not.

Reproducibility. The property that another person, given your data and your script, obtains exactly your numbers.

The advantage of the script over the menu is permanence: you write it once and rerun it on every new dataset, and anyone can audit exactly what was done. This is the same seven-step procedure of Procedure 4.2; Python simply executes the computational steps instantly, while steps 1–2 (hypotheses, α) and the final conclusion in context remain in your head and in your report.

Procedure 10.2 — Running a hypothesis test in software.

1. Formulate H_0 and H_1 (4.1) and choose α *before* touching the software.
2. Select the test with Procedure 4.5 (one sample) or Procedure 5.2 (two samples), and verify its assumptions on the data (normality check, independence, sample-size conditions).
3. Run the matching command (Tables 10.4 and 10.5) with the hypothesized value and, if offered, the correct direction of H_1.
4. Read the test statistic and the p-value. If the software reported a two-sided p-value and your H_1 is one-sided, apply (10.1).
5. Check the statistic by hand from the summary statistics in the same output, and check that the CI and the test decision agree.
6. State the conclusion in the language of the problem, with the effect size, not only ''reject/fail to reject.''

Engineering judgment. With $n =$ 10,000 observations, even a trivial difference becomes ''statistically significant'' at $p < 0.001$. Software makes huge samples easy, which makes this trap common. Always report the estimated effect and its interval alongside the p-value, and ask whether the effect matters at engineering scale.

Reference. Montgomery & Runger, 6th ed., Sections 10-1 to 10-6.

10.5 *Two-Sample Methods in Software*

Chapter 5 taught four different two-sample t procedures, and choosing among them was the hard part (Procedure 5.2). In Minitab the choice is encoded in menus and checkboxes, and Table 10.5 is the complete map.

Test	Minitab menu path	Key option	Implements
Two-sample t (pooled)	`Stat` → `Basic Statistics` → `2-Sample t`	Check "Assume equal variances"	(5.4)
Two-sample t (Welch)	`2-Sample t`	Uncheck equal variances	(5.6), (5.7)
Paired t	`Paired t`	Two matched columns	(5.8)
Two variances (F-test)	`2 Variances`	Levene's test also shown	(5.13)
Two proportions	`2 Proportions`	Event counts and trials	(5.10)

Table 10.5. The two-sample menu map. Every path starts from `Stat` → `Basic Statistics`.

For tensile strength measured on two production lines, the pooled test produces:

```
Minitab --- 2-Sample t: Line1 vs Line2 (pooled)

Sample    N    Mean   StDev   SE Mean
Line1    10   48.50    2.30      0.73
Line2    12   47.10    2.50      0.72

Difference: μ1 − μ2    Estimate: 1.400
95% CI for difference: (−0.334, 3.134)

Null hypothesis        H0: μ1 − μ2 = 0
Alternative hypothesis H1: μ1 − μ2 ≠ 0

T-Value   DF   P-Value
   1.68   20     0.108

Both use Pooled StDev = 2.4166
```

Everything Chapter 5 computed in five steps arrives in one block: the pooled standard deviation $s_p = 2.4166$ from (5.3), the statistic $T_0 = 1.68$ from (5.4) with $n_1 + n_2 - 2 = 20$ degrees of freedom, the exact p-value, and the CI for $\mu_1 - \mu_2$ from (5.5). Since $p = 0.108 > 0.05$ we fail to reject H_0, and consistently, the CI includes 0.

A useful working habit: run `2 Variances` *first*. If the F-test of (5.13) rejects equal variances, uncheck "Assume equal variances" in the `2-Sample t` dialog, and Minitab switches to the Welch statistic (5.6) and computes the fractional Satterthwaite degrees of freedom (5.7) for you—a computation that is tedious by hand and free by machine.

Example 10.2 — Port dwell times: a pooled test in Python

A logistics analyst compares mean container dwell times (hours) at two terminals. Terminal A: $n_1 = 12$, $\bar{x}_1 = 61.30$, $s_1 = 8.20$. Terminal B: $n_2 = 14$, $\bar{x}_2 = 54.60$, $s_2 = 7.60$. Sample variances are similar, and normal probability plots show no problems, so the pooled test is appropriate. The analyst runs

```
t_stat, p_value = stats.ttest_ind(termA, termB, equal_var=True)
print(f"T = {t_stat:.2f}, p = {p_value:.3f}")
```

and the script prints `T = 2.16, p = 0.041`. Verify the statistic by hand, state the decision at $\alpha = 0.05$, and give the 95 % CI for the difference.

Solution.

Step 1 — Pooled variance.

By (5.3),

$$s_p^2 = \frac{11(8.20)^2 + 13(7.60)^2}{12 + 14 - 2} = \frac{739.6 + 750.9}{24} = 62.10, \qquad s_p = 7.881.$$

Step 2 — Test statistic.

By (5.4),

$$T_0 = \frac{61.30 - 54.60}{7.881\sqrt{\frac{1}{12} + \frac{1}{14}}} = \frac{6.700}{7.881 \times 0.3934} = \frac{6.700}{3.100} = 2.161,$$

matching the printed `T = 2.16` with 24 degrees of freedom.

Step 3 — Decide.

The exact p-value $0.041 < 0.05$: **reject** H_0. The t table could only have said $0.02 < p < 0.05$.

Step 4 — Interval.

With $t_{0.025,24} = 2.064$, (5.5) gives $6.700 \pm 2.064 \times 3.100 = 6.700 \pm 6.398$, i.e. $(0.30,\ 13.10)$ hours. The interval excludes 0, consistent with the rejection. The answer is $\boxed{T_0 = 2.16,\ p = 0.041;\ \text{reject } H_0;\ \text{CI } (0.30,\ 13.10) \text{ h}}$.

Startup lens. Every A/B testing dashboard is a two-proportion z-test (5.10) running behind a friendly interface. When a dashboard declares a "winner," ask it the questions of Procedure 10.2: what were the hypotheses, was the p-value one-sided or two-sided, and how many times did you peek at the data before stopping?

Reference. Montgomery & Runger, 6th ed., Sections 11-1 to 11-8 and 12-1 to 12-6.

10.6 *Regression in Software*

Regression is where software becomes indispensable. Fitting (6.2) by hand required the sums S_{xx} and S_{xy} of (6.7) and (6.8); fitting the multiple regression model (7.1) by hand requires inverting a matrix (7.4), which is impractical beyond two predictors. Minitab's `Stat` → `Regression` → `Fit Regression Model` does both, and prints three blocks: a coefficients table, a model summary, and an ANOVA table. Figure 10.3 shows a complete output for a study of product strength versus process temperature ($n = 20$), annotated field by field with the equation each number implements.

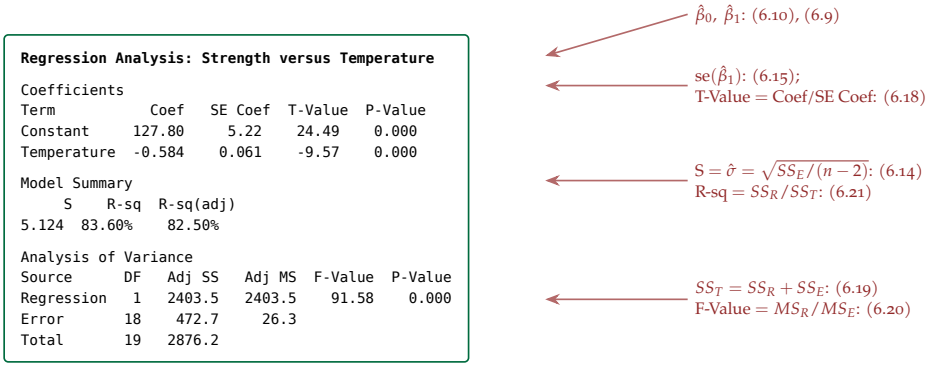

Figure 10.3. An annotated Minitab regression output. Every printed field is an equation from Chapter 6.

Read the output in this order:

1. **Coefficients.** The fitted line is $\hat{y} = 127.80 - 0.584x$: each degree of temperature costs 0.584 units of strength. The row for `Temperature` carries the slope test of (6.18): $T_0 = -0.584/0.061 = -9.57$ with p-value 0.000 (meaning $p < 0.0005$), so the slope is significant.

2. **Model Summary.** `S` is the estimated residual standard deviation $\hat{\sigma} = \sqrt{26.3} = 5.124$ from (6.14); `R-sq` is $R^2 = 2403.5/2876.2 = 83.6\,\%$ from (6.21).

3. **Analysis of Variance.** The decomposition (6.19) appears as the SS column, and $F_0 = 2403.5/26.3 = 91.58$ is the regression F-test of (6.20).

For simple linear regression the coefficient t-test and the ANOVA F-test are the same test, and the output lets you check the identity:

$$T_0^2 = F_0 \qquad \text{(simple linear regression only)}, \tag{10.2}$$

here $(-9.57)^2 = 91.58$. If a printed output violates (10.2), something is wrong with the output—or it is not a simple linear regression.

Insight. The identity $T_0^2 = F_0$ is a free integrity check, and it illustrates the deeper point of this chapter: because you know the formulas behind the fields, you can *cross-examine* an output. Analysts who know only which cell to read cannot tell a healthy printout from a corrupted one.

Four-in-one residual plot. Minitab's standard 2×2 panel of residual diagnostics: normal probability plot, residuals versus fits, histogram, and residuals versus observation order.

The `Graphs` button in the regression dialog produces the *four-in-one* residual plot, which automates Procedure 6.4. Figure 10.4 shows the four panels for a healthy fit. Your job is interpretation: a curved normal probability plot signals non-normality; a funnel shape in residuals-versus-fits signals non-constant variance; a drift or cycle in residuals-versus-order signals dependence between observations.

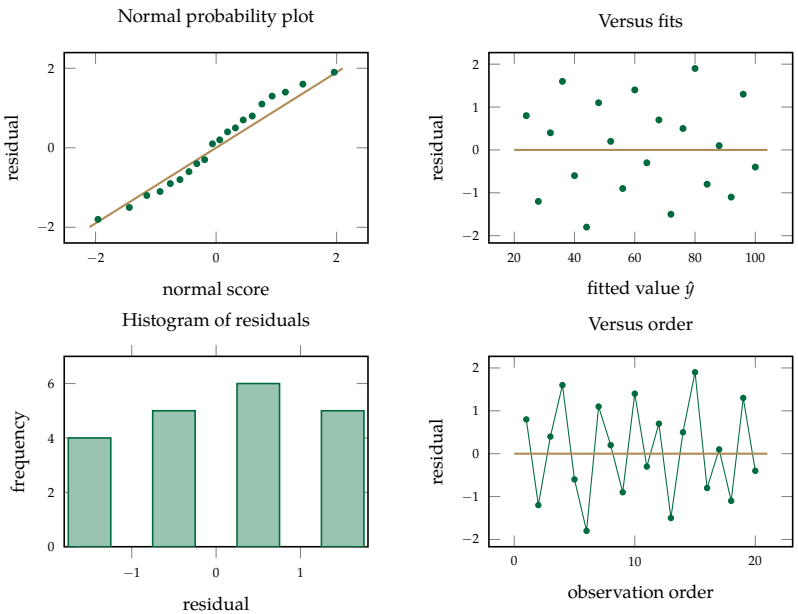

Figure 10.4. The four-in-one residual plot for a healthy regression fit: the probability plot is straight (normality), the versus-fits panel is a structureless band (constant variance), the histogram is roughly symmetric, and the versus-order panel shows no drift (independence).

For multiple regression the same command accepts several predictor columns. The coefficients table then carries one row per predictor, each with the individual

t-test of (7.12); the model summary adds `R-sq(adj)`, the adjusted R^2 of (7.11), which unlike R^2 can *decrease* when a useless predictor is added; the ANOVA block carries the overall F-test of (7.9); and requesting VIFs adds the variance inflation factors of (7.17) for multicollinearity screening. In Python, `stats.linregress(x, y)` covers simple linear regression (returning the slope, intercept, r, the slope p-value, and the slope standard error), and the `statsmodels` package's `ols` covers multiple regression with a Minitab-like summary table.

Procedure 10.3 — Fitting and reading a regression in software.

1. Plot the data first (scatter plot). Software will fit a line to any cloud, including one that is obviously curved.
2. Fit the model: Minitab `Stat` → `Regression` → `Fit Regression Model`; Python `stats.linregress` (one predictor) or `statsmodels` `ols` (several).
3. Read the coefficients table: fitted equation, then each term's t-test (6.18), (7.12) against $H_0 : \beta_j = 0$.
4. Read the model summary: $\hat{\sigma}$ (the field `S`), R^2 (6.21), and for multiple regression the adjusted R^2 (7.11).
5. Read the ANOVA block: the decomposition (6.19) and the overall F-test (6.20), (7.9). For simple linear regression, confirm $T_0^2 = F_0$ (10.2).
6. Examine the four-in-one residual plot (Procedure 6.4) before trusting any interval or prediction from the model.

Example 10.3 — Pricing experiment: reading a regression output

A startup varies its subscription price across $n = 20$ landing-page cohorts and records the signup conversion rate (%). Minitab returns:

```
Regression Analysis: Conversion versus Price

Coefficients
```

```
Term          Coef   SE Coef   T-Value   P-Value
Constant    12.640     0.842     15.01     0.000
Price      −0.0525    0.0098     −5.36     0.000

Model Summary

    S    R-sq   R-sq(adj)
1.204   61.5%       59.3%

Analysis of Variance

Source       DF   Adj SS   Adj MS   F-Value   P-Value
Regression    1    41.65    41.65     28.73     0.000
Error        18    26.09     1.45
Total        19    67.74
```

(a) Write the fitted line and interpret the slope. (b) Is price a significant predictor at $\alpha = 0.01$? (c) Verify `S`, `R-sq`, and the identity (10.2). (d) Predict conversion at a price of 120 SAR.

Solution.

Step 1 — Fitted line.

From the Coef column, $\hat{y} = 12.640 - 0.0525x$. Each 1 SAR of price reduces expected conversion by about 0.0525 percentage points; equivalently, a 20 SAR price increase costs about 1.05 points of conversion.

Step 2 — Slope test.

The Price row gives $T_0 = -0.0525/0.0098 = -5.36$ with $p < 0.0005 < 0.01$: reject H_0: $\beta_1 = 0$ (6.18). Price is a significant predictor.

Step 3 — Cross-checks.

From the ANOVA block, $\hat{\sigma} = \sqrt{MS_E} = \sqrt{1.45} = 1.204 =$ `S` (6.14); $R^2 = SS_R/SS_T = 41.65/67.74 = 0.615$ (6.21), matching `R-sq`; and $T_0^2 = (-5.36)^2 = 28.7 = F_0$, confirming (10.2).

Step 4 — Prediction.

At $x = 120$, $\hat{y} = 12.640 - 0.0525 \times 120 = 12.640 - 6.300 = 6.340$. The answer is $\boxed{\hat{y} = 12.640 - 0.0525x;\ \hat{y}(120) \approx 6.34\%}$. For a decision about a specific future cohort, ask the software for the prediction interval (6.27) at $x_0 = 120$, which is wider than the CI for the mean response (6.26).

Startup lens. Notice what the regression does *not* say: it does not say 120 SAR is the profit-maximizing price. Revenue per visitor is price times conversion, and tracing that curve from the fitted line is the founder's model to build. Software fits the line; the pricing decision is yours.

Reference. Montgomery & Runger, 6th ed., Sections 13-1 to 13-4 and 14-3 to 14-5.

10.7 *ANOVA and Designed Experiments in Software*

The experimental designs of Chapters 8 and 9 follow the same pattern: the software builds the ANOVA table you learned to build by hand, and adds exact p-values and multiple-comparison output. Table 10.6 is the map.

Analysis	Minitab menu path	Python	Implements
One-way ANOVA	`Stat` → `ANOVA` → `One-Way`	`f_oneway`	(8.11)
Tukey pairwise	`One-Way` → `Comparisons` → `Tukey`	`tukey_hsd`	(8.15)
Fisher LSD	`One-Way` → `Comparisons` → `Fisher`	pairwise t with (8.14)	(8.14)
RCBD	`Stat` → `ANOVA` → `Balanced ANOVA`	`statsmodels` `anova_lm`	(8.24)
Two-factor ANOVA	`Stat` → `ANOVA` → `General Linear Model`	`statsmodels` `anova_lm`	(9.10), (9.11)

Table 10.6. Method-to-tool map for the designed experiments of Chapters 8 and 9.

Three reading rules cover all of these outputs. First, the rows of the software ANOVA table are exactly the decomposition you learned: (8.5) for one-way, (8.22) for the RCBD, and (9.3) with an interaction row for the two-factor design. Second, each F-Value is the ratio of that row's mean square to the error mean square, as in (8.11). Third, for a two-factor output, read the *interaction* row first (9.10): if the interaction is significant, the main-effect p-values on the rows above it cannot be interpreted separately, no matter how small they are.

Procedure 10.4 — Running an ANOVA in software.

1. Arrange the data in stacked form: one column for the response, one column for each factor (and one for blocks in an RCBD). Menu software expects this layout.
2. Run the matching command from Table 10.6.
3. Check the degrees-of-freedom column against the design: $a-1$ for a factor with a levels, $b-1$ for blocks, $(a-1)(b-1)$ for the interaction. Wrong degrees of freedom mean the software misread your layout.

4. Read the F-statistic and exact p-value for each effect; for two-factor designs, interaction first (9.10).
5. If the factor is significant and has more than two levels, request Tukey comparisons (8.15) to find *which* means differ.
6. Examine the residual plots (same four-in-one panel as regression) before reporting.

Example 10.4 — Comparing guardrail configurations: one-way ANOVA and Tukey

A red team measures the false-positive rate (blocked benign prompts per 1,000) for four guardrail configurations A–D, with $n = 6$ runs each. Minitab's `One-Way` with Tukey comparisons prints:

```
One-way ANOVA: FPrate versus Config

Source   DF   Adj SS   Adj MS   F-Value   P-Value
Config    3   204.84    68.28      7.76     0.001
Error    20   176.00     8.80
Total    23   380.84

Tukey Pairwise Comparisons (95%)

Grouping Information

Config   N    Mean   Grouping
D        6   31.00   A
B        6   30.20   A
C        6   25.10     B
A        6   24.50     B

Means that do not share a letter are significantly different.
```

(a) Verify the F-Value and state the ANOVA decision at $\alpha = 0.05$. (b) Compute the Tukey critical distance ($q_{0.05}(4, 20) = 3.96$) and confirm the grouping letters.

Solution.

Step 1 — ANOVA.

The mean squares are $MS_{\text{Treat}} = 204.84/3 = 68.28$ and $MS_E = 176.00/20 = 8.80$ (8.8), (8.9), so by (8.11) $F_0 = 68.28/8.80 = 7.76$, matching the output. The exact p-value 0.001 is far below 0.05: **reject** equal means. (The F table would only say $p < 0.01$, since $f_{0.01,3,20} = 4.94 < 7.76$.)

Step 2 — Tukey distance.

By (8.15),

$$\text{HSD} = q_{0.05}(4, 20)\sqrt{\frac{MS_E}{n}} = 3.96\sqrt{\frac{8.80}{6}} = 3.96 \times 1.211 = 4.80.$$

Step 3 — Pairwise reading.

Differences larger than 4.80 are significant: $|\text{D} - \text{A}| = 6.50$, $|\text{D} - \text{C}| = 5.90$, $|\text{B} - \text{A}| = 5.70$, and $|\text{B} - \text{C}| = 5.10$ all exceed 4.80; $|\text{D} - \text{B}| = 0.80$ and $|\text{C} - \text{A}| = 0.60$ do not. So {B, D} and {A, C} form two internally indistinguishable groups—exactly the two grouping letters printed. The answer is

$F_0 = 7.76,\ p = 0.001$; configurations A and C are the low-false-positive group.

Safety lens. The grouping letters are the deployable conclusion: any configuration in the low group is statistically defensible, so the final choice between A and C can be made on other grounds—latency, cost, red-team coverage—without pretending the data ranked them.

Reference. Compiled from Lectures 5–26; the equation references point to the chapters of this book.

10.8 The Complete Method-to-Tool Map

Table 10.7 assembles the whole course in one place: every method from Chapters 2 through 9, the Minitab path and Python function that run it, and the book equation it implements. Keep it beside you when you work; it is the dictionary between this book and any output you will read in practice.

Method	Minitab	Python	Equation
Point estimates $\bar{x}$, s	`Basic Statistics → Display Descriptive Statistics`	`np.mean`, `np.std(ddof=1)`	(1.1), (1.2)
CI for μ (σ known)	`1-Sample Z`	`norm.ppf`	(3.1)
CI for μ (σ unknown)	`1-Sample t`	`t.ppf`, `t.interval`	(3.5)
CI for σ^2	`1 Variance`	`chi2.ppf`	(3.6)
CI for p	`1 Proportion`	`norm.ppf`	(3.7)
z-test on μ	`1-Sample Z`	`norm.cdf`	(4.3)
t-test on μ	`1-Sample t`	`ttest_1samp`	(4.6)
χ^2-test on σ^2	`1 Variance`	`chi2.cdf`	(4.7)
z-test on p	`1 Proportion`	`proportions_ztest`	(4.8)
Two means, σ's known	summary formula	`norm.cdf`	(5.1)
Pooled t	`2-Sample t`, equal variances checked	`ttest_ind(..., equal_var=True)`	(5.4)
Welch t	`2-Sample t`, equal variances unchecked	`ttest_ind(..., equal_var=False)`	(5.6)
Paired t	`Paired t`	`ttest_rel`	(5.8)
Two variances	`2 Variances`	`f.cdf`	(5.13)
Two proportions	`2 Proportions`	`proportions_ztest`	(5.10)
Simple linear regression	`Regression → Fit Regression Model`	`linregress`	(6.9), (6.18)
Multiple regression	`Fit Regression Model`	`statsmodels ols`	(7.4), (7.9)
One-way ANOVA	`ANOVA → One-Way`	`f_oneway`	(8.11)
Tukey comparisons	`One-Way → Comparisons`	`tukey_hsd`	(8.15)
RCBD	`ANOVA → Balanced ANOVA`	`statsmodels anova_lm`	(8.24)
Two-factor ANOVA	`ANOVA → General Linear Model`	`statsmodels anova_lm`	(9.10)

Table 10.7. The complete method-to-tool map for ISE 315. Minitab paths start from the `Stat` menu; Python functions are in `scipy.stats` except `proportions_ztest`, `ols`, and `anova_lm`, which are in `statsmodels`, and the NumPy estimators in the first row.

The course also provides a Jupyter notebook (Blackboard → Course Materials → Supplementary) that wraps the one- and two-sample methods of Chapters 3 through 5: you upload raw data as a CSV file or type summary statistics, select the test from a menu, and the notebook returns the test statistic, the exact p-value, the confidence interval, and a plain-English conclusion. It is intended for homework verification and for your own projects; exams are done by hand with tables.

Reference. Montgomery & Runger, 6th ed., Appendix A tables, compared with the computer outputs shown in the text.

10.9 *Rounding: Tables versus Software*

Software answers and table answers will disagree in the last digit or two, and you should be able to explain why. There are three sources.

1. **Table granularity.** The appendix t, χ^2, and F tables print a handful of tail probabilities. A hand computation can therefore only *bracket* a p-value ($0.02 < p < 0.05$), while software integrates the density and reports $p = 0.035$. Both describe the same shaded area in Figure 10.2; the table merely reads it through

a coarser grid.

2. **Rounded critical values.** Tables print $t_{0.025,24} = 2.064$; the exact quantile is 2.0639 . . . An interval computed by hand with the printed value can differ from the software interval in the third decimal. This difference is cosmetic, never decisive.

3. **Intermediate rounding by you.** If you round $s_p = 2.4166$ to 2.42 midway through a two-sample computation, your T_0 can drift by a few hundredths. Carry four significant figures through intermediate steps, as this book does, and hand and machine answers will agree to the digits that matter.

A caution specific to Excel: its statistical functions (for example `T.INV` and `CHISQ.INV`) have historically differed slightly from dedicated statistical packages in extreme tails, and its rounded display can hide precision loss. Excel is fine for data entry and quick checks; use Minitab or Python when the p-value is close to α.

Please pay attention to what rounding can and cannot do: it can never flip a clear decision ($p = 0.001$ stays a rejection under any rounding), but a p-value within about 0.01 of α deserves exact computation and, more importantly, an honest sentence in the report acknowledging that the evidence is marginal.

Insight. Exactness is not the same as validity. The p-value 0.035 is exact *given the model*: normal population, independent observations, correctly chosen test. A violated assumption perturbs the p-value by far more than any table rounding ever will. Precision in the third decimal is worthless when the first decimal rests on an unchecked assumption.

Reference. Lecture 16; the course policy on AI tools is in the syllabus.

AI assistant. A large-language-model tool (chat or code assistant) that can draft statistical analyses, write `scipy` code, and explain output—fluently, and sometimes wrongly.

10.10 *Using AI Assistants Responsibly*

AI assistants have joined Minitab and Python in the engineer's toolbox: you can paste in data and ask for ''a test of whether these two samples differ,'' and receive working code and a confident narrative in seconds. Used well, they are a fast way to draft code and to get a first explanation of an unfamiliar output field. Used carelessly, they add a new failure mode on top of every pitfall in this chapter:

the assistant chooses the test for you, invisibly, and defends its choice fluently whether or not it is right.

The risks are specific and checkable:

- **Silent test selection.** Asked to ''compare before and after,'' an assistant may run an independent two-sample t-test on paired data—the exact error that Procedure 5.2 exists to prevent—and report a plausible p-value.
- **Unstated assumptions.** The generated code rarely checks normality, independence, or the proportion conditions $n\hat{p} \geq 5$ and $n(1-\hat{p}) \geq 5$ unless you ask.
- **Confident arithmetic errors.** Assistants sometimes state numerical results (a T_0, a critical value) without actually running code. A number that was not produced by executed code must be verified.
- **Direction and defaults.** The same one-sided/two-sided trap of (10.1) applies, now hidden inside generated code you may not read.

The defense is the knowledge you already have. Treat an AI assistant the way a senior engineer treats an eager junior analyst: assign clearly, demand the reasoning, and check the work.

Procedure 10.5 — Using an AI assistant for statistical work.

1. State the problem fully in your prompt: how the data were collected (paired or independent, sample sizes), what is known (σ known or not), the direction of the engineering question, and your α.
2. Ask the assistant to *name the test* it proposes and to *state that test's conditions* (normality, independence, sample size). If it cannot state the conditions, do not use its answer.
3. Check the named test against this book: does it match Procedure 4.5 or Procedure 5.2? Does the code call the function that Table 10.7 pairs with that test?

4. Run the code yourself and verify one anchor number by hand (the standard error, or the test statistic from summary statistics), exactly as in Procedures 10.1 and 10.2.
5. Ask for the assumption checks (residual plots, normality check) and interpret them yourself. The assistant's verdict on its own plots is not evidence.
6. Write the final interpretation in your own words, in the language of the engineering problem. Responsibility for the conclusion is not delegable.

Safety lens. The verification-and-validation habit transfers directly: an AI assistant is an unvalidated component in your analysis pipeline. You do not have to prove it is always right—you have to check its output on the case in front of you, against procedures you trust, before the result enters a decision.

Engineering judgment. A useful prompt pattern: after the assistant answers, ask ''what would have to be true about my data for this analysis to be wrong?'' A correct analysis survives the question; a wrong one usually confesses its missing assumption in the first sentence of the reply.

Chapter Summary

Software executes every method of this course faster and more precisely than hand computation: it produces exact p-values instead of table brackets, computes interval bounds and awkward quantities like Welch degrees of freedom (5.7) and the regression matrix solution (7.4), and draws the diagnostic plots that Procedure 6.4 needs. What it never does is choose the method, check the assumptions, or interpret the result—the division of labor of Table 10.2. This chapter's working skills are: running and hand-checking intervals (Procedure 10.1), tests (Procedure 10.2), regressions (Procedure 10.3), and ANOVAs (Procedure 10.4); converting two-sided p-values to one-sided ones correctly (10.1); using the identity $T_0^2 = F_0$ (10.2) as an output integrity check; explaining table-versus-software rounding differences; and supervising AI assistants with Procedure 10.5. The master dictionary between methods, tools, and equations is Table 10.7.

Output field	Book quantity	Equation
`SE Mean`	$s/\sqrt{n}$ or $\sigma/\sqrt{n}$	(2.5)
`T-Value` (one sample)	$(\bar{x}-\mu_0)/(s/\sqrt{n})$	(4.6)
`Pooled StDev`	s_p	(5.3)
`P-Value`	exact tail probability of the statistic	(4.4)
One-sided from two-sided	$p_{\text{two}}/2$ or $1-p_{\text{two}}/2$	(10.1)
`Coef` / `SE Coef`	$\hat{\beta}_j$ and $\text{se}(\hat{\beta}_j)$	(6.9), (6.15)
`S` (regression)	$\hat{\sigma}=\sqrt{SS_E/(n-2)}$	(6.14)
`R-sq`, `R-sq(adj)`	R^2, adjusted R^2	(6.21), (7.11)
`F-Value` (regression)	MS_R/MS_E, and $T_0^2=F_0$ in SLR	(6.20), (10.2)
`F-Value` (ANOVA)	MS_{Treat}/MS_E	(8.11)

Table 10.8. Software output fields at a glance: what each printed number is, in the notation of this book.

Exercises

Which tool, which method

Exercise 10.1. For each scenario, name the statistical method, the Minitab menu path, and the Python function (use Table 10.7). (a) A perception team has 23 measured localization errors and wants a 95 % CI for the mean error; σ is unknown. (b) A warehouse compares picking times for two independent groups of workers using two different scanners; the two sample variances are similar. (c) A startup measures page-load time on the same 14 servers before and after a caching change. (d) A fleet manager tests whether the variance of fuel use exceeds a contractual limit σ_0^2.

Exercise 10.2. An engineer has $n = 40$ dwell-time measurements and wants a 95 % CI for the mean. Fifteen years of terminal records establish $\sigma = 7.2$ hours, and there is no reason to believe the spread has changed. A colleague says: ''Just use `1-Sample t`; it is the safer default since it does not assume σ.'' Is the colleague's advice harmful, harmless, or helpful? Explain what changes in the output and in the interval width.

Reading output

Exercise 10.3. A warehouse analyst runs `1-Sample t` on 16 picking times (seconds) and receives:

```
 N     Mean   StDev   SE Mean   95% CI for μ
16   48.250   6.400     1.600   (44.840, 51.660)
```

(a) Verify the `SE Mean` field. (b) Recover the multiplier the software used and confirm it is $t_{0.025,15}$, not $z_{0.025}$. (c) Write the interval conclusion in context.

Exercise 10.4. A churn analysis regresses monthly customer churn (%) on onboarding support hours across $n = 18$ customer cohorts:

```
Term          Coef   SE Coef   T-Value   P-Value
Constant     4.820     0.610      7.90     0.000
Hours       −0.312     0.085     −3.67     0.002

Source       DF   Adj SS   Adj MS   F-Value   P-Value
Regression    1    11.90    11.90     13.47     0.002
Error        16    14.14    0.884
Total        17    26.04
```

(a) Write the fitted line and test $H_0\colon \beta_1 = 0$ at $\alpha = 0.01$. (b) Verify the T-Value from the Coef and SE Coef fields and check the identity (10.2). (c) Compute R^2 and $\hat{\sigma}$ from the ANOVA block.

Exercise 10.5. A red-team study compares mean defects found per session across three testing protocols ($n = 6$ sessions each). The printed one-way ANOVA table arrived smudged; fill in the four missing entries and state the decision at $\alpha = 0.05$ ($f_{0.05,2,15} = 3.68$).

```
Source     DF   Adj SS   Adj MS   F-Value   P-Value
Protocol    ?    98.40        ?         ?     0.005
Error      15        ?     6.30
Total      17   192.90
```

Exam-style problems

Exercise 10.6. A transportation team compares delivery lateness (minutes) under two routing algorithms, with independent samples and similar variances. Minitab prints:

```
Sample    N    Mean   StDev   SE Mean
Alg1     15   12.80    3.10      0.80
Alg2     15   10.40    2.90      0.75

Difference: μ1 − μ2    Estimate: 2.400
95% CI for difference: (0.155, 4.645)

T-Value   DF   P-Value
   2.19   28     0.037

Both use Pooled StDev = 3.0017
```

(a) **[3 pt]** State the hypotheses and name the exact procedure the software ran, citing the option that selected it. (b) **[5 pt]** Verify the Pooled StDev and the T-Value by hand. (c) **[3 pt]** State the decision at $\alpha = 0.05$ two ways: from the p-value and from the CI. (d) **[3 pt]** The team's real question was one-sided: does Algorithm 2 *reduce* mean lateness? Give the one-sided p-value and the decision.

Exercise 10.7. An evaluation team monitors the *consistency* of an LLM's benchmark scores. The contract requires score variance $\sigma^2 = 9$; excessive variance triggers re-tuning. From $n = 15$ runs, $s = 4.20$. Software output:

```
Null hypothesis        H0: σ² = 9
Alternative hypothesis H1: σ² ≠ 9
Method: Chi-Square

Statistic   DF   P-Value
    27.44   14     0.034
```

(a) **[4 pt]** Verify the statistic by hand. (b) **[3 pt]** What could the χ^2 table have told you, given $\chi^2_{0.025,14} = 26.12$ and $\chi^2_{0.01,14} = 29.14$? Compare with the software p-value. (c) **[3 pt]** State the decision at $\alpha = 0.05$ and one assumption that must hold for this p-value to be trustworthy.

Assessing outside recommendations

Exercise 10.8. A founder asks an AI assistant to check whether a redesign improved signup conversion on the company's 12 regional landing pages, measured before and after the redesign on the *same* 12 pages. The assistant replies: ''I ran `scipy.stats.ttest_ind(before, after)` and got $p = 0.21$. There is no significant improvement.'' The founder notices that every single page improved, most by a small amount, while pages differ hugely from one another (some convert at 2%, some at 11%). Diagnose the assistant's error, explain why the p-value came out large, name the correct analysis, and predict qualitatively what it will show.

Exercise 10.9. A consultant evaluates whether a new scheduling system reduced mean truck turnaround time at a depot ($H_1 : \mu_{\text{new}} < \mu_{\text{old}}$). The software reported a two-sided p-value of 0.056, and the consultant writes: ''Since our test is one-sided, the p-value is $0.056/2 = 0.028 < 0.05$, so the reduction is significant.'' The summary statistics in the same output show $\bar{x}_{\text{new}} = 41.8$ and $\bar{x}_{\text{old}} = 40.1$ minutes. Find the error, compute the correct one-sided p-value, and state the correct conclusion.

In-Class Activity 10.1: Reading the Printout

Week 8 — Lecture 16

Work in teams of 3–4. Write your reasoning, not just the final number. One submission per team, all names on it.

A logistics company piloted a route-optimization system on 10 trucks, measuring each truck's fuel use (L/100 km) for one month *before* and one month *after* the change. An analyst hands your team this printed output and says ''the software says it works.''

Paired T: Before, After

Sample	N	Mean	StDev	SE Mean
Before	10	33.40	4.10	1.30
After	10	31.90	3.90	1.23
Difference	10	1.500	1.580	0.500

95% lower bound for μ_D: 0.584

Null hypothesis H0: μ_D = 0
Alternative hypothesis H1: μ_D > 0

T-Value	P-Value
3.00	0.007

Part 1 — What test is this, and why?

Name the test, write the hypotheses in symbols, and explain in one or two sentences why the paired design is right here—and what large source of variation it removes compared with treating the 20 numbers as two independent samples.

Part 2 — Check the machine

Using only the `Difference` row, verify the `SE Mean` and the `T-Value` by hand. Then state the decision at $\alpha = 0.05$ and what the ''95 % lower bound 0.584'' adds to the story.

Part 3 — What did the software not *check?*

List two assumptions this analysis needs that appear nowhere in the printout, and for each, say
your team would check it (a plot, a follow-up question about how the data were collected, etc

Exit check

One sentence: in the division of labor between engineer and software, which single step of th
only have been done by the engineer?