Quantitative Research Methods

Chapter 0b: Precourse Toolkit

Prof. Dr. Christoph Weisser

HSBI

Summer Semester 2026

The course at a glance

	Lecture	Topic
	0	Precourse (a) — statistics refresher: description, probability, inference
▶	**0b**	**Precourse (b) — notation, logs and odds, likelihood, Python**
	1	Ch. 1 + 2 — Introduction; what is statistical learning
	2	Ch. 2 — Accuracy, bias–variance, Bayes classifier, KNN
	3–4	Ch. 3 — Linear regression (estimation, inference, diagnostics)
	5–6	Ch. 4 — Classification (logistic, LDA/QDA, naive Bayes, ROC)
	7	Ch. 5 — Resampling (validation, CV, bootstrap)
	8	Ch. 6 — Model selection and regularisation (ridge, lasso)
	9	Ch. 7 — Moving beyond linearity (splines, GAMs)
	10	Ch. 8 — Tree-based methods (trees, forests, boosting)
	11	Ch. 10 — Deep learning (MLPs, CNNs, training)
	12	Ch. 13 — Multiple testing (FWER, FDR)

The second half of the optional precourse. Part (a) refreshed the *statistics*; this session covers the *language and machinery* the later chapters use without explanation.

Contents

Optional and advanced material is collected in the **appendix** at the end of the deck.

Why this second session exists

These five topics were counted in the ten lecture decks. Each is used *heavily* and explained *nowhere* — the chapters assume you already read them fluently.

Topic	Where it bites	Uses
log and exp	Ch. 4 (113), Ch. 10 (31), Ch. 6 (16)	176
odds and the logit	Ch. 4 (92), Ch. 10 (12)	108
likelihood and $\prod$	Ch. 4 (35), Ch. 2–3	37
$\sum$, arg max, indicator, set notation	every chapter from Ch. 2	180
counting and 2^p cost	Ch. 6 (subset selection)	13

The claim

None of this is hard. All of it is *unfamiliar notation for familiar ideas* — and unfamiliar notation is what makes a lecture feel impossible to follow.

Notation introduced in this session

Symbol	Meaning
$\sum_{i=1}^{n} a_i$	add up $a_1, \ldots, a_n$
$\prod_{i=1}^{n} a_i$	multiply $a_1, \ldots, a_n$
$\arg\min_\theta f(\theta)$	the *value of* θ that makes f smallest
$I(\text{condition})$	1 if the condition holds, 0 otherwise
$x \in \mathcal{S}$, $\lvert\mathcal{S}\rvert$	x is in the set $\mathcal{S}$; the size of $\mathcal{S}$
log, exp	natural logarithm (base e) and its inverse e^x
$\text{odds} = \frac{p}{1-p}$	how much more likely than not
$\text{logit}(p) = \log \frac{p}{1-p}$	log-odds: a probability on an unbounded scale
$\sigma(z) = \frac{1}{1+e^{-z}}$	the logistic (sigmoid) function; the inverse of the logit
$L(\theta)$, $\ell(\theta)$	likelihood and log-likelihood of the parameter θ
$\hat{\theta}_{\text{MLE}}$	the value of θ that maximises the likelihood
$\binom{p}{k}$	the number of ways of choosing k items from p
2^p	the number of subsets of p predictors: how the cost grows

Learning objectives for this session

By the end you will be able to:

- **Read aloud** any formula in the course: sums, products, arg max, indicators, subscripts and set notation.
- **Manipulate** logs and exponentials, and explain what a coefficient means when the response is logged.
- **Convert** freely between probability, odds and log-odds, and explain why classification models work on the logit scale.
- **Write down** a likelihood, take its log, and explain what maximum likelihood chooses — and why it coincides with least squares under normal errors.
- **Count** models and explain why 2^p makes exhaustive search hopeless.
- **Write** the Python the labs actually use: functions, loops, comprehensions, seeds, and the `fit`/`predict` pattern.

Where this chapter is used in industry

Sector	Concrete application	Which decision it drives
Banking	Credit scorecards and probability-of-default models, built on the log-odds scale	Approve, refer or decline — and at what price
Retail, airlines	Log–log demand models, whose coefficient *is* the price elasticity	How far a price is allowed to move
Insurance	Frequency and severity models fitted by maximum likelihood	The technical premium quoted for a policy
Finance	Continuously compounded log returns; CAGR	What gets reported as performance and volatility
Telco, SaaS	Retention curves that decay roughly exponentially	How much may be spent to acquire a customer
Operations, call centres	Arrival counts and service times fitted by Poisson and exponential likelihoods	Staffing levels set for each interval

In industry — The common thread

In each row the model is linear on a *transformed* scale — log, log-odds — and someone must transform back before the number is used.

Industry case in depth: price elasticity of demand

A retailer or an airline needs one number: how much volume a price rise costs.

In industry — The model

Response: log(units sold), per product and week. **Predictors**: log(price) plus controls — a promotion flag, seasonality, a competitor price index, the channel. **Why both sides are logged**: the slope of a log–log regression *is* the elasticity — read straight off the output, nothing to transform back.

Illustrative

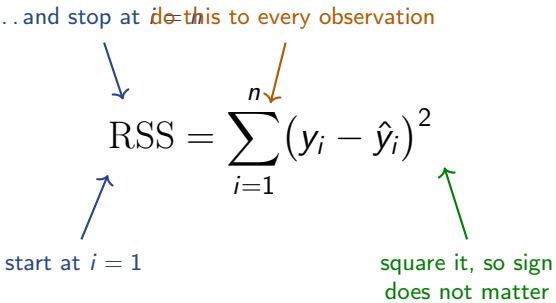
An estimated $\beta_1 \approx -1.8$ says a 1% price rise costs about 1.8% of volume. For a bigger move the multiplicative form is the honest one: a 10% price cut lifts volume by $0.9^{-1.8} - 1 \approx 21\%$ (not 18%), so revenue *rises*. Comparing $|\beta_1|$ with 1 is the whole calculation: above 1 demand is elastic and a price rise loses revenue; below 1 it gains.

Who signs off — and the honest caveat

Pricing or category management own the list price, promotion depth and fare-class prices; finance checks the margin. The number itself lands with the category manager setting next season's price ladder. Caveat: past prices were cut *because* demand was weak, so the estimate is confounded unless it comes from experiments or genuinely exogenous price moves — and never extrapolate to a price never tried.

Outline

The anatomy of a formula



How to read any formula

Read it as an *instruction*, left to right: "for each observation, take the gap between what happened and what we predicted, square it, and add those up". A formula is a recipe, not a hieroglyph.

Sums and products

The two operators that appear everywhere

$$\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n, \qquad \prod_{i=1}^{n} a_i = a_1 \times a_2 \times \cdots \times a_n.$$

With numbers

For $a = (2, 5, 3)$: $\sum_i a_i = 10$ and $\prod_i a_i = 30$. In Python: `a.sum()` and `a.prod()`.

Rules worth knowing

$$\sum_i (a_i + b_i) = \sum_i a_i + \sum_i b_i, \qquad \sum_i c\, a_i = c \sum_i a_i, \qquad \sum_{i=1}^{n} c = nc.$$

The one that is *not* a rule

$\sum_i a_i b_i \neq \left(\sum_i a_i\right)\left(\sum_i b_i\right)$, and $\sum_i a_i^2 \neq \left(\sum_i a_i\right)^2$.

arg min, arg max and the indicator

min versus arg min

$\min_\theta f(\theta)$ is the smallest *value* of f; $\arg\min_\theta f(\theta)$ is the θ *that achieves it*.

$$f(\theta) = (\theta - 3)^2 + 5 \quad \Longrightarrow \quad \min_\theta f = 5, \qquad \arg\min_\theta f = 3.$$

Estimation is always an arg min (of a loss) or an arg max (of a likelihood): we want the *parameter*, not the value of the criterion.

The indicator function

$$I(\text{condition}) = \begin{cases} 1 & \text{if the condition is true,} \\ 0 & \text{otherwise.} \end{cases}$$

It turns counting into arithmetic. The classification error rate is

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \quad — \quad \text{“the fraction of observations we got wrong”.}$$

Sets, subscripts and the notation of a data set

Sets

$x \in \mathcal{S}$ means "x belongs to $\mathcal{S}$"; $|\mathcal{S}|$ is the number of elements in $\mathcal{S}$. In Chapter 8, R_j is the set of observations falling in region j of a tree, and $\frac{1}{|R_j|}\sum_{i \in R_j} y_i$ is just "the mean response of the points in that box".

Subscripts and superscripts

x_{ij}	observation i, variable j (row first, then column)
x_i	the whole i-th row (a vector of p numbers)
β_j	the coefficient belonging to variable j
$\hat{y}_i^{(-i)}$	prediction for i from a model fitted *without* i
α^{*b}	the estimate from bootstrap sample b (a label, not a power)

Read the superscript before you square it

In Chapter 5, $\hat{\alpha}^{*b}$ is "the estimate from resample b". A superscript is often an *index*, not an exponent — context decides.

Exercise 0b.1 — Translating notation [Math]

Exercise

Let $y = (3, 1, 4)$ and $\hat{y} = (2, 2, 4)$, so $n = 3$.

1. Write $\sum_{i=1}^{3}(y_i - \hat{y}_i)^2$ in words, then compute it.
2. Compute $\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$ and say what it measures.
3. For $g(\theta) = (\theta - 4)^2 + 7$, give $\min_\theta g$ and $\arg\min_\theta g$.
4. Let $\mathcal{S} = \{i : y_i > 2\}$. List $\mathcal{S}$, give $|\mathcal{S}|$, and compute $\frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} y_i$.
5. Is $\sum_i (y_i - \hat{y}_i)^2$ the same as $\left(\sum_i (y_i - \hat{y}_i)\right)^2$? Check with these numbers.

Solution — Exercise 0b.1

Solution

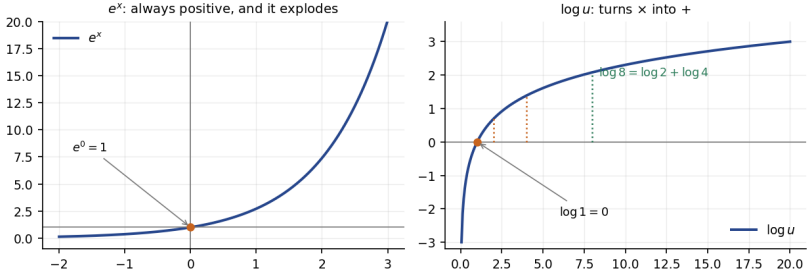
(1) "For each of the three observations, subtract the prediction from the observation, square the result, and add the three numbers." Differences: $1, -1, 0$; squares $1, 1, 0$; total 2.
(2) The indicator is 1 for observations 1 and 2 (where $y_i \neq \hat{y}_i$) and 0 for the third, so the average is $2/3 \approx 0.67$: *the fraction of predictions that are wrong* — the misclassification rate of Chapters 2 and 4.
(3) A square is never negative, so g is smallest when $(\theta - 4)^2 = 0$. Hence $\arg\min_\theta g = 4$ and $\min_\theta g = 7$. Note the two answers are different objects: one is a parameter, the other a loss value.
(4) $\mathcal{S} = \{1, 3\}$ (observations with $y_i > 2$), so $|\mathcal{S}| = 2$ and the mean over the set is $(3 + 4)/2 = 3.5$. This is exactly the calculation a regression tree does inside one leaf.
(5) No. Here $\sum_i (y_i - \hat{y}_i)^2 = 2$ but $\left(\sum_i (y_i - \hat{y}_i)\right)^2 = (1 - 1 + 0)^2 = 0$. Squaring first keeps the errors from cancelling — which is the whole reason least squares squares.

Common mistake

Writing min when arg min is meant. Every estimator ahead is an arg min: $\hat{\beta}$ is the *coefficient value* that minimises RSS, never the RSS value itself.

Outline

The two functions Chapter 4 is written in



The rules, all of them

$$\log(ab) = \log a + \log b, \quad \log(a/b) = \log a - \log b, \quad \log(a^k) = k \log a,$$
$$\log 1 = 0, \quad \log e = 1, \quad e^{\log a} = a, \quad \log(e^x) = x, \quad e^{a+b} = e^a e^b.$$

"log" in this course always means the *natural* logarithm, base $e \approx 2.718$ — what `np.log` computes.

Why logs are everywhere in statistics

Three jobs a log does

1. **Turns multiplying into adding.** A likelihood is a product over n observations; its log is a sum, which we can differentiate. (Next section.)
2. **Turns unbounded ratios into a symmetric scale.** Odds run from 0 to ∞; log-odds run from $-\infty$ to ∞. (Section after that.)
3. **Tames skew and turns effects into percentages.** A right tail is compressed, and a coefficient on $\log y$ reads as a *relative* change.

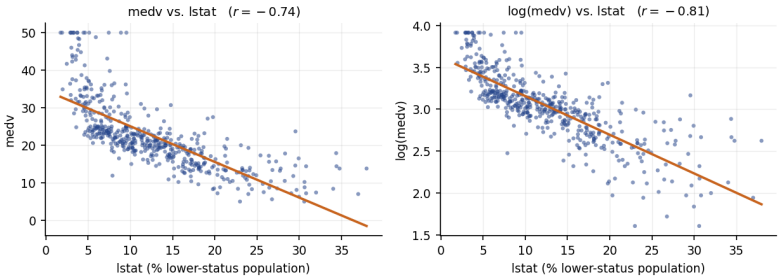
Numerical underflow — the practical reason

A likelihood multiplies n probabilities. With $n = 1000$ and each around 0.5, the product is $\approx 10^{-301}$, close to the smallest number a computer can store; at $n = 2000$ it underflows to zero and every model looks equally good. Its log ≈ -693 is perfectly ordinary.

In industry — Finance and pricing: why logs are everywhere

Log returns add up over time; and in a log–log demand model the slope *is* the price elasticity.

A log scale can straighten a relationship



`Boston`: median house value against the share of lower-status population. On the raw scale the cloud curves and $r = -0.74$; after logging the response it is close to straight, and $r = -0.81$.

How to read a logged coefficient

In $\log y = \beta_0 + \beta_1 x$, a one-unit rise in x multiplies y by e^{β_1}. For small β_1, that is roughly a $100\beta_1\%$ change: $\beta_1 = -0.05$ means "about 5% lower".

Exercise 0b.2 — Logs in practice [Math]

Exercise

1. Simplify $\log(x^3 y) - \log(y)$ and $\log\left(e^{2x}\right)$.
2. Solve $e^{0.4t} = 5$ for t.
3. A model gives $\log(\text{wage}) = 2.1 + 0.08\,\text{educ}$. By what factor does an extra year of education multiply the wage? Express it as a percentage.
4. Prices rise 3% a year. Write the price after t years, take logs, and say why the log of the price is a straight line in t.
5. Why can you not take the log of a variable that contains zeros, and what do people do instead?

Solution — Exercise 0b.2

Solution

(1) $\log(x^3 y) - \log y = \log x^3 + \log y - \log y = 3 \log x$. And $\log(e^{2x}) = 2x$, since log and exp undo each other.
(2) Take logs of both sides: $0.4t = \log 5 = 1.609$, so $t = 1.609/0.4 = 4.02$.
(3) An extra year adds 0.08 to log(wage), so it multiplies the wage by $e^{0.08} = 1.083$: about +8.3%. Note the shortcut "$0.08 \to 8\%$" is an approximation that is good for small coefficients and poor for large ones ($e^{0.5} = 1.65$, not 1.5).
(4) $P_t = P_0(1.03)^t$, so $\log P_t = \log P_0 + t \log(1.03) = \log P_0 + 0.0296\, t$. Constant *growth rates* are straight lines on a log scale — which is why growth data are always plotted that way.
(5) $\log 0 = -\infty$ (and logs of negatives are undefined), so a single zero destroys the column. Common fixes: model $\log(y + 1)$; use a square root instead; or — best — use a model built for the data type, such as the Poisson regression of Chapter 4 for counts.

Common mistake

Inventing a rule for sums. $\log(a + b)$ simplifies to nothing — the log rules turn *products* into sums; there is no rule for the log of a sum.

Outline

Three ways to say the same thing

Definitions

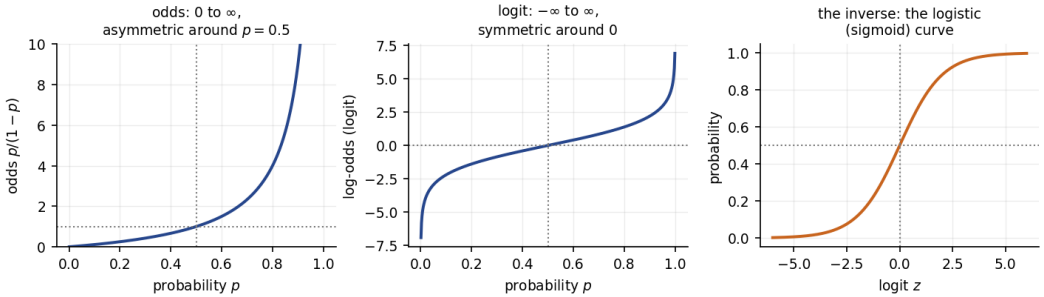
$$\text{odds} = \frac{p}{1-p}, \qquad \text{logit}(p) = \log \frac{p}{1-p}, \qquad p = \sigma(z) = \frac{1}{1+e^{-z}}.$$

probability	odds	log-odds	said aloud
0.10	0.11	-2.20	"1 to 9 against"
0.25	0.33	-1.10	"1 to 3 against"
0.50	1.00	0.00	"evens"
0.75	3.00	$+1.10$	"3 to 1 on"
0.90	9.00	$+2.20$	"9 to 1 on"

Why the log-odds scale

Probabilities are squeezed against 0 and 1; log-odds run from $-\infty$ to $+\infty$, symmetrically about zero. Models work on the log-odds scale because it has *room*: whatever number a linear predictor produces is a legal log-odds.

The three scales, drawn

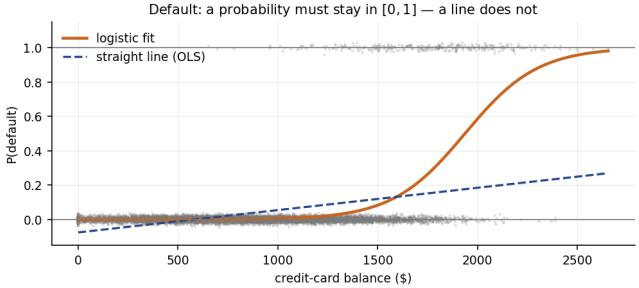


Left to right: probabilities become odds (0 to ∞), odds become log-odds ($-\infty$ to ∞), and the logistic function $\sigma(z) = 1/(1 + e^{-z})$ carries them back. Because σ can never leave $(0, 1)$, *any* number a linear model produces becomes a legal probability.

In industry — Betting and insurance: odds are not probabilities

A bookmaker's price is odds; converting back gives implied probabilities that sum to more than one — the margin. Read odds as probabilities and you misprice the risk.

Why Chapter 4 cannot just fit a straight line



`Default` ($n = 10\,000$, 3.3% default). The dashed line is ordinary least squares on a 0/1 response: it predicts *negative* probabilities for small balances and would exceed 1 eventually. The logistic curve cannot.

The one-line summary of logistic regression

Model the *log-odds* as linear: $\log \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x$, then convert back with σ. That is the whole idea; Chapter 4 fills in the details.

Exercise 0b.3 — Odds and odds ratios [Math]

Exercise

In a promotion study, 45% of customers who received a voucher bought something, against 25% of those who did not.

1. Compute the odds of buying in each group.
2. Compute the **odds ratio** (voucher vs. no voucher) and state in words what it says.
3. Compute the log-odds in each group, and check that their difference equals the log of the odds ratio.
4. A logistic model reports a coefficient of 0.90 for the voucher dummy. Convert it to an odds ratio.
5. If the log-odds of an outcome are -1.4, what is the probability?

Solution — Exercise 0b.3

Solution

(1) Voucher: $0.45/0.55 = 0.818$. No voucher: $0.25/0.75 = 0.333$.
(2) Odds ratio $= 0.818/0.333 = 2.45$: the *odds* of buying are about two and a half times higher with a voucher. Note this is not "2.45 times as likely" — the ratio of *probabilities* is only $0.45/0.25 = 1.8$. Odds ratios and risk ratios are different numbers, and confusing them overstates effects.
(3) $\log 0.818 = -0.201$ and $\log 0.333 = -1.099$; the difference is $0.898 = \log(2.45)$ ✓. *Differences on the log-odds scale are ratios on the odds scale* — which is exactly what a logistic coefficient is.
(4) $e^{0.90} = 2.46$ — the same odds ratio, as it must be: the coefficient of a dummy *is* the log odds ratio.
(5) $p = \sigma(-1.4) = 1/(1 + e^{1.4}) = 1/(1 + 4.055) = 0.198$, about 20%.

Common mistake

Reading odds as probabilities. Odds of 0.818 do not mean "82%" — convert back with $p = \text{odds}/(1 + \text{odds}) = 0.45$.

Outline

The idea in one sentence

Maximum likelihood

Among all parameter values, choose the one that makes the data you actually observed *most probable*.

The likelihood function

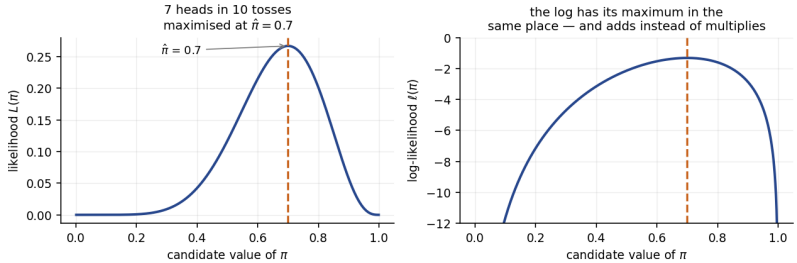
For independent observations $y_1, \ldots, y_n$ with parameter θ,

$$L(\theta) = \prod_{i=1}^{n} P(y_i \mid \theta), \qquad \ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log P(y_i \mid \theta).$$

Read the direction of the conditioning

$P(y_i \mid \theta)$ is a probability *of the data given a parameter*. As a function of θ with the data *fixed*, we call it a likelihood — and it is **not** a probability distribution over θ. That is the commonest misunderstanding here.

A coin, worked end to end



Seven heads in ten tosses. The likelihood $L(\pi) = \binom{10}{7}\pi^7(1-\pi)^3$ peaks at $\hat{\pi} = 0.7$; the log-likelihood peaks in the same place, because log is increasing.

Why we always maximise the log

Same answer, easier algebra (sums, not products), and no numerical underflow. Every "log-likelihood" in Chapters 4, 6 and 10 is this.

Maximum likelihood outside the classroom

In industry — Insurance and banking: pricing models

Claim frequency and severity models, probability-of-default models and demand models are nearly all fitted by maximum likelihood rather than least squares — because the response is a count, a 0/1 event, or a skewed positive amount, and none of those has normal errors.

In industry — Telco and software: exponential decay

Retention curves fall away roughly exponentially, so the log of the share still active is close to linear in time. Fitting on that scale gives the expected customer lifetime that acquisition budgets and lifetime-value figures rest on.

What one principle buys

The same estimation idea covers continuous responses, classes, counts and durations — which is why Chapters 4, 6 and 10 all report a log-likelihood instead of an R^2.

Exercise 0b.4 — What a likelihood is and is not [Concept]

Exercise

Decide whether each statement is correct, and correct the ones that are not.

1. "The likelihood that $\pi = 0.7$ is 0.27, so there is a 27% chance the true value is 0.7."
2. "The maximum-likelihood estimate is the most probable value of the parameter."
3. "Adding an observation always lowers the likelihood."
4. "Maximising the likelihood and maximising the log-likelihood can give different answers."
5. "A model with a higher log-likelihood always predicts new data better."

Solution — Exercise 0b.4

Solution

(1) Wrong. A likelihood is a probability of the *data*, computed *given* a parameter value — not a probability of the parameter. Likelihoods need not sum or integrate to 1 over π, so "27% chance" is meaningless here. (Turning it into a statement about π requires a prior and Bayes' theorem.)
(2) Wrong, in the same way. It is the value *under which the observed data are most probable*. The distinction matters exactly when someone asks "how sure are you?" — that question is answered by a standard error or an interval, not by the likelihood's height.
(3) Correct for discrete data. Each extra observation multiplies in another probability ≤ 1, so L shrinks; for a continuous response the factors are densities and can exceed 1. Only *comparisons at the same data set* are meaningful — which is why you can never compare log-likelihoods across samples of different size.
(4) Wrong. log is strictly increasing, so it preserves the location of the maximum. The maximiser is identical; only the value differs.
(5) Wrong. Likelihood is measured on the *training* data, and a more flexible model can always be made to fit it better — overfitting. That is precisely why Chapter 6 penalises the log-likelihood (AIC, BIC) and Chapter 5 measures performance on held-out data instead.

Common mistake: the root of (1), (2) and (4) is one flipped conditioning: L is the probability of *this data* given π, never of π given the data. Keep that direction straight and the statements sort themselves.

Outline

Functions and loops — used in all fifteen labs

```
def rmse(y, y_hat): # define once, reuse everywhere
    """Root mean squared error."""
    return np.sqrt(np.mean((y - y_hat) ** 2))

scores = [] # accumulate results in a list
for degree in range(1, 6): # range(1, 6) -> 1, 2, 3, 4, 5
    fit = np.polyfit(x_train, y_train, degree)
    scores.append(rmse(y_test, np.polyval(fit, x_test)))

best = int(np.argmin(scores)) + 1 # argmin gives the POSITION
print(f"best degree: {best} (RMSE {min(scores):.3f})")
```

The pattern behind every tuning experiment

Define a scoring function; loop over candidate settings; store each score; take the `argmin`. Chapters 5–8 change what happens inside the loop, never the shape of it.

Four idioms worth ten minutes each

```
squares = [d ** 2 for d in range(5)] # list comprehension
mask = df["balance"] > 1000 # a boolean Series...
df.loc[mask, "income"].mean() # ...used to select rows

for name, group in df.groupby("education"): # iterate over groups
    print(name, group["wage"].median())

for ax, col in zip(axes.ravel(), columns): # zip pairs things up
    ax.hist(df[col])
```

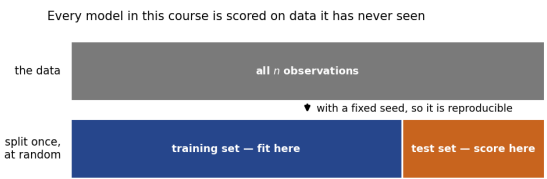
Two things that will bite you

1. **Indexing starts at 0**, and `range(1, 6)` *excludes* 6. Off-by-one errors are the most common lab bug.
2. **Chained assignment**: `df[df.x > 1]["y"] = 0` silently does nothing. Use `df.loc[df.x > 1, "y"] = 0`.

Randomness, seeds and reproducibility

```
rng = np.random.default_rng(2024) # ONE generator, seeded once
sample = rng.choice(data, size=100)

from sklearn.model_selection import train_test_split
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.3,
                                          random_state=42)
```



A "random" generator is a deterministic recipe and the seed is where it starts, so the same seed replays the same draws — which is why your numbers match the slides. Change the seed and the third decimal moves: not a bug, but sampling variability, which Chapter 5 is about measuring.

The scikit-learn contract: fit, predict, score

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model = LinearRegression() # 1. choose a model
model.fit(X_train, y_train) # 2. learn from TRAINING data only
pred = model.predict(X_test) # 3. predict on unseen data
mean_squared_error(y_test, pred) # 4. score honestly
model.coef_, model.intercept_ # fitted values carry a trailing _
```

Learn it once and you know every model in the course

Trees, forests, boosting, SVMs and neural nets use the same three methods: swapping `LinearRegression()` for `RandomForestRegressor()` changes one line.

The cardinal sin

Never call `fit` on the test set — not for the model, and not for a scaler either (Chapter 5).

Exercise 0b.6 — Reading and fixing code [Python]

Exercise

A student writes the following to compare polynomial degrees:

```
scores = []
for degree in range(1, 5):
    fit = np.polyfit(x, y, degree) # (a)
    scores.append(rmse(y, np.polyval(fit, x))) # (b)
best = np.argmin(scores) # (c)
print("best degree:", best)
```

1. Which degrees are actually tried?
2. Line (c) prints 3 for the best degree. Why is that answer wrong, and what is the correct degree?
3. Lines (a) and (b) use the same data. What will `scores` look like as the degree rises, and why does that make the comparison meaningless?
4. Rewrite the loop so the comparison is honest. Name the one function you need.
5. Why should the code set a seed?

Solution — Exercise 0b.6

Solution

(1) `range(1, 5)` yields $1, 2, 3, 4$ — the upper limit is *excluded*. Degree 5 is never tried.
(2) `np.argmin` returns a *position*, and positions start at 0. Position 3 is the fourth entry, i.e. degree 4. The fix is `best = np.argmin(scores) + 1`, or better, store the degrees alongside the scores.
(3) The model is scored on the same data it was fitted to, so the RMSE can only fall as the degree rises — a high enough polynomial passes through every point and scores 0. The loop therefore always "discovers" the most complex model. This is overfitting, the central problem of Chapter 2.
(4) Split first, fit on training data, score on test data:

```
X_tr, X_te, y_tr, y_te = train_test_split(x, y, test_size=.3, random_state=1)
for degree in range(1, 6):
    fit = np.polyfit(X_tr, y_tr, degree)
    scores.append(rmse(y_te, np.polyval(fit, X_te)))
```

The function is `sklearn.model_selection.train_test_split` (Chapter 5 replaces it with cross-validation).
(5) Without a seed the split differs on every run, so the "best" degree changes between runs and neither you nor a reader can reproduce the number. A seed makes the result checkable — it does not make it correct.

Common mistake: the thread through all five bugs — trusting a printed number without asking what was computed. Off-by-one and leakage errors run *silently*; print intermediate objects and check them.

Extended Exercise 0b.2 — A complete small experiment [Python]

Extended exercise (15 min)

You are given `Auto` and want to know how well `horsepower` predicts `mpg`, and whether a curve beats a line.

1. Write, in words or code, the six steps of the experiment from loading the data to reporting a number.
2. Which step must come *before* any model is fitted, and why?
3. On one 70/30 split you get test RMSEs 4.96, 4.52, 4.54, 4.55, 4.44 for degrees 1 to 5. Which degree do you choose, and which part of that sequence is real?
4. Rerunning with 30 different seeds, degree 5 wins 21 times, degree 2 six times and degree 3 three times. What does that tell you about your answer to (3), and what would you do about it?
5. Your colleague standardised `horsepower` using the mean and SD of the *whole* data set before splitting. Explain the problem in one sentence.

Solution — Extended Exercise 0b.2

Solution

(1) The six steps. Load; *look* (scatter, and check for missing values — `Auto` stores them as `'?'`); split with a seed; fit each candidate on train; score each on test; report the chosen model *with* its test error.
(2) Looking, and splitting. Plot before modelling — the curvature is obvious and says a polynomial is worth trying. The split must precede every fitting decision, so nothing learned from the test set can influence the model.
(3) Only the first step is real. The drop from 4.96 to 4.52 is large and reflects genuine curvature: a line is the wrong shape. Beyond that the numbers differ by less than 0.12 — ordinary split-to-split noise. Degree 5 is nominally lowest, but nothing meaningful separates it from degree 2 — so choose **degree 2**, the simplest model that is not beaten.
(4) The winner is unstable, so do not trust it. A single split is a high-variance estimate, and the seed sweep proves it: three different "best" degrees appear. The fix is k-fold cross-validation (Chapter 5): here (5-fold, repeated 10×) it gives RMSEs of 4.91, 4.36, 4.37, 4.38, 4.34 with a repeat-to-repeat spread of about ± 0.04: degree 1 is clearly worse, and *degrees 2 to 5 are statistically indistinguishable*. The defensible sentence is "a quadratic is a real improvement on a line; beyond that this data set cannot tell".
(5) Leakage. The scaler learned its mean and SD from data that includes the test set, so the test error is no longer an honest estimate of performance on new data. Fit the scaler on the training set alone.
Common mistake: rerunning with new seeds until the ranking looks right — the sweep is a diagnostic, not a menu; a fixed seed buys reproducibility, not correctness.

Outline

The session in one slide

- **Notation** is instructions: $\sum$ add, $\prod$ multiply, $\arg\min$ "the value that minimises", $I(\cdot)$ a 0/1 switch, $|\mathcal{S}|$ a count. Read formulas aloud.
- **Logs** turn products into sums, ratios into differences and skew into symmetry; e^{β} converts a logged coefficient back into a multiplicative effect.
- **Odds** $= p/(1-p)$; **logit** $=$ log odds, on $(-\infty, \infty)$; σ brings it back to $(0, 1)$. Logistic regression is a linear model *on the logit scale*.
- **Likelihood** is the probability of the observed data as a function of the parameter; maximise its log. Under normal errors this is least squares; under Bernoulli it gives k/n and, with predictors, logistic regression.
- **Counting** explains Chapter 6: 2^p subsets is hopeless, $p^2/2$ is not.
- **Python**: a function, a loop, a seed, and `fit`/`predict` on a split — that is every lab.

Key formulas at a glance

Notation

$$\sum_{i=1}^{n} a_i, \quad \prod_{i=1}^{n} a_i, \quad \arg\min_{\theta} f(\theta), \quad \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i).$$

Logs, odds and the logistic function

$$\log(ab) = \log a + \log b, \quad \log(a^k) = k \log a, \quad \log \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x \implies \text{odds ratio} = e^{\beta_1}.$$

$$\text{odds} = \frac{p}{1-p}, \quad p = \frac{\text{odds}}{1+\text{odds}}, \quad \text{logit}(p) = \log \frac{p}{1-p}, \quad \sigma(z) = \frac{1}{1+e^{-z}}.$$

Likelihood and counting

$$\ell(\theta) = \sum_{i=1}^{n} \log P(y_i \mid \theta), \qquad \hat{\theta} = \arg\max_{\theta} \ell(\theta), \qquad \binom{p}{k}, \quad \sum_{k=0}^{p} \binom{p}{k} = 2^p.$$

Vocabulary check

Term	One-line meaning
Summation / product notation	compact instructions to add or multiply
Indicator function	a 0/1 switch that turns counting into arithmetic
arg min / arg max	the input that minimises / maximises, not the value
Natural logarithm	the inverse of e^x; `np.log`
Odds	$p/(1-p)$: how much more likely than not
Odds ratio	ratio of two odds — what a logistic coefficient exponentiates to
Logit	log-odds; the scale on which classification is linear
Logistic / sigmoid	$\sigma(z) = 1/(1+e^{-z})$; maps any number into $(0, 1)$
Likelihood	probability of the observed data, as a function of the parameter
Log-likelihood	its log: sums instead of products, no underflow
Maximum likelihood	the parameter that makes the observed data most probable
Cross-entropy	$-\ell$ for a classifier; the loss of Chapter 10
Binomial coefficient	$\binom{p}{k}$: the number of subsets of size k
Data leakage	letting test information reach the training pipeline
Seed	fixes the random draw so results are reproducible

Where each topic reappears in the course

Topic from this session	Comes back in
$\sum$, $\prod$, $\arg\min$, $I(\cdot)$	every chapter; RSS and error rates from Ch. 2
Sets and $\lvert\mathcal{S}\rvert$	Ch. 8 (tree regions), Ch. 12 (clusters)
Logs and exponentials	Ch. 3 (logged responses), Ch. 4, Ch. 10
Odds, logits, the sigmoid	Ch. 4 (logistic regression), Ch. 10 (activations)
Likelihood and log-likelihood	Ch. 4 (fitting), Ch. 6 (AIC, BIC)
Maximum likelihood	Ch. 4, Ch. 6, Ch. 10 (cross-entropy)
2^p and counting	Ch. 6 (subset selection)
Loops, functions, `argmin`	every lab; tuning loops in Ch. 5–8
Seeds and splits	Ch. 2 onwards; formalised in Ch. 5
`fit` / `predict`	every model in the course
Leakage	Ch. 5 (the rule), Ch. 6 (scaling inside CV)

Common pitfalls to leave behind

Don't

1. Confuse min with arg min, or a value with an index.
2. Read $\hat{\alpha}^{*b}$ as a power — superscripts are often labels.
3. Write $\sum_i a_i^2 = (\sum_i a_i)^2$. It is false, and it is the reason squared errors work.
4. Report an odds ratio as if it were a ratio of probabilities.
5. Say "the likelihood that the parameter is 0.3". Likelihood is about the *data*, given the parameter.
6. Take logs of a variable containing zeros without saying what you did.
7. Score a model on the data it was fitted to.
8. Forget the seed — or change it until the answer looks better.

Self-check questions

1. Read $\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$ aloud, in words. (pp. 10 and 12)
2. What is the difference between $\min_\theta f$ and $\arg\min_\theta f$? (p. 12)
3. Simplify $\log(a^2 b) - 2\log a$. (p. 17)
4. A logistic coefficient is -0.7. What happens to the odds? (p. 23)
5. Convert log-odds $+1.6$ into a probability. (p. 23)
6. Why is a log-likelihood a sum when a likelihood is a product? (p. 30)
7. Under what assumption is least squares equivalent to maximum likelihood? (appendix, p. 53)
8. How many models does best-subset selection fit with $p = 20$? (appendix, p. 57)
9. What does `np.argmin` return — and what does it not return? (p. 12)
10. Name two things that must never be fitted on the test set. (p. 38)

Five things to remember

1. **A formula is a recipe.** Read it aloud as an instruction and it stops being intimidating.
2. **Logs convert.** Products to sums, ratios to differences, growth to straight lines, coefficients to percentages.
3. **Classification lives on the logit scale**, because probabilities do not have room to be linear.
4. **Maximum likelihood is one principle** behind least squares, logistic regression, AIC/BIC and cross-entropy.
5. **Fit on training data, score on test data, seed everything.** Every lab, every chapter, no exceptions.

What's next

Lecture 1 — *Chapter 1: Introduction* and the start of *Chapter 2: Statistical Learning*.

- $Y = f(X) + \varepsilon$: what statistical learning is trying to do.
- Prediction versus inference; supervised versus unsupervised.
- The course data sets and the notation that goes with them.

Companion notebook

`Lab_Notebooks/chapter_00b_lab.ipynb` runs everything in this deck: the logit conversions (§3), a likelihood maximised by hand and by `scipy` (§4), the 2^p count (§5), and a complete split/fit/score experiment on `Auto` (§6, *The Python patterns every lab uses*).

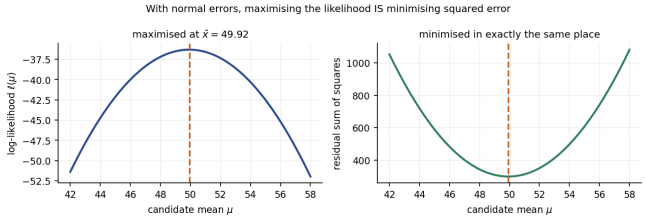
Appendix — what is in here, and why

Nothing in the appendix is needed to follow the chapter: each slide is a formal derivation, a longer worked exercise, or a side topic. Read it when you want the full story, or when a later chapter sends you back here.

Contents of the appendix

- **Least squares as maximum likelihood** — the derivation that connects Chapter 3 to Chapter 4, plus Extended Exercise 0b.1 (deriving an estimator from scratch). The two methods are usable without it.
- **Counting and computational cost** — how the 2^p models of Chapter 6 explode, and why "just try everything" fails (with Exercise 0b.5). Chapter 6 re-states the conclusion where it is needed.

Least squares is maximum likelihood in disguise



With normal errors $P(y_i \mid \mu) = \frac{1}{\sigma\sqrt{2\pi}} \exp\!\left(-\frac{(y_i-\mu)^2}{2\sigma^2}\right)$, and the log strips the exponential down to its exponent: $\log P(y_i \mid \mu) = -\frac{(y_i-\mu)^2}{2\sigma^2} + \text{const}$, so

$$\ell(\mu) = -\frac{1}{2\sigma^2}\sum_i (y_i - \mu)^2 + \text{const} \quad \Longrightarrow \quad \arg\max_\mu \ell(\mu) = \arg\min_\mu \sum_i (y_i - \mu)^2.$$

One principle under two names

Maximising the likelihood *is* minimising squared error whenever the errors are normal. Chapter 4 keeps the principle and drops the normal assumption — that is all logistic regression does.

Extended Exercise 0b.1 — Deriving an estimator [Math]

Extended exercise (15 min)

In a sample of n customers, k default. Model each customer as an independent Bernoulli trial with unknown default probability π.

1. Write the likelihood $L(\pi)$ as a product, then simplify it to a single expression in π, k and n.
2. Take logs to get $\ell(\pi)$.
3. Differentiate ℓ with respect to π, set the derivative to zero, and solve. What is $\hat{\pi}_{\mathrm{MLE}}$?
4. Evaluate it for the `Default` data: $n = 10\,000$ with 333 defaults.
5. Explain why maximising ℓ gives the same answer as maximising L, and why we bother with the log at all.
6. Chapter 4 replaces the single π by $\pi(x_i) = \sigma(\beta_0 + \beta_1 x_i)$. Write the resulting log-likelihood, and say why it can no longer be solved by hand.

Solution — Extended Exercise 0b.1 (1/2)

Solution

(1) The likelihood. Each defaulting customer contributes π and each non-defaulting one $1-\pi$; independence lets us multiply:

$$L(\pi) = \prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i} = \pi^{k}(1-\pi)^{n-k}, \qquad k = \sum_i y_i.$$

The exponent trick is worth noticing: $\pi^{y_i}(1-\pi)^{1-y_i}$ equals π when $y_i = 1$ and $1-\pi$ when $y_i = 0$.

(2) The log-likelihood.

$$\ell(\pi) = k \log \pi + (n-k)\log(1-\pi).$$

The product has become a sum — the only reason this is doable by hand.

(3) Maximise.

$$\frac{d\ell}{d\pi} = \frac{k}{\pi} - \frac{n-k}{1-\pi} = 0 \implies k(1-\pi) = (n-k)\pi \implies \hat{\pi}_{\mathrm{MLE}} = \frac{k}{n}.$$

The sample proportion — exactly what anyone would have guessed. That is the point: maximum likelihood *derives* the obvious estimator, and keeps working when the answer is no longer obvious.

Solution — Extended Exercise 0b.1 (2/2)

Solution

(4) The numbers. $\hat{\pi} = 333/10\,000 = 0.0333$, the 3.3% default rate quoted in Chapter 4.
(5) Why the log is safe — and necessary. log is strictly increasing, so it preserves the location of the maximum: if $L(a) > L(b)$ then $\ell(a) > \ell(b)$. We use it because it converts the product into a sum (so the derivative is a sum of simple terms) and because the product itself underflows to zero on a computer for even moderate n.
(6) The regression version. With $\pi_i = \sigma(\beta_0 + \beta_1 x_i)$,

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \Big[y_i \log \pi_i + (1 - y_i) \log(1 - \pi_i) \Big].$$

Setting the derivatives to zero now gives equations in which β appears *inside* σ, so they cannot be rearranged into a closed form as the normal equations were. Software maximises ℓ numerically — by walking uphill, exactly the gradient method of Part (a). Note also that $-\ell$ is the "binary cross-entropy" loss of Chapter 10: the same quantity under a different name.

Common mistake

Treating $\hat{\pi} = k/n$ as exact. With $k = 0$ the MLE announces "defaults are impossible" — an estimate inherits the sample's noise and needs a standard error, not blind faith.

How many models are there?

Counting rules

$$k! = k(k-1)\cdots 1, \qquad \binom{p}{k} = \frac{p!}{k!\,(p-k)!}, \qquad \sum_{k=0}^{p} \binom{p}{k} = 2^p.$$

$\binom{p}{k}$ counts the subsets of size k; summing over all k counts *every* subset — each variable is either in or out, hence 2^p.

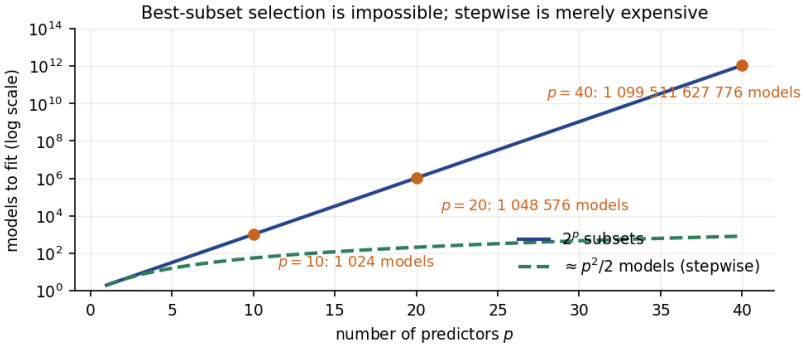
With numbers

With $p = 4$ predictors there are $\binom{4}{2} = 6$ two-variable models and $2^4 = 16$ models in total (including the empty one). With $p = 10$: 1 024. With $p = 40$: over 10^{12}.

Where you meet this

Chapter 6 asks which subset of predictors to keep. The count above is why "try them all" stops being possible around $p = 30$, and why stepwise selection and regularisation exist at all.

The cost of "just try everything"



Forward stepwise selection fits about $p^2/2$ models instead of 2^p: for $p = 40$, roughly 800 instead of 10^{12}. At 0.01 s per fit that is eight seconds against about 350 years — with no guarantee of finding the same answer, which is the trade-off Chapter 6 discusses.

Exercise 0b.5 — Counting [Math]

Exercise

A data set has $p = 6$ candidate predictors.

1. How many models contain exactly two predictors?
2. How many models are there in total?
3. Best-subset selection fits all of them; forward stepwise fits $1 + p(p+1)/2$. Compute both counts.
4. Each fit takes 0.01 seconds. How long does each strategy take for $p = 6$, and for $p = 30$?
5. Testing every model against the data and keeping the best R^2 has a statistical cost as well as a computational one. What is it? (One sentence; Chapter 13 gives it a name.)

Solution — Exercise 0b.5

Solution

(1) $\binom{6}{2} = \frac{6 \cdot 5}{2} = 15$.
(2) $2^6 = 64$ (the empty model included).
(3) Best subset: 64 fits. Forward stepwise: $1 + 6(7)/2 = 22$ fits. For $p = 30$ the same formulas give $2^{30} \approx 1.07 \times 10^9$ against 466.
(4) At 0.01 s per fit and $p = 6$: 0.64 s versus 0.22 s — no practical difference. At $p = 30$: 1.07×10^7 seconds $\approx$ 124 days versus 4.7 seconds. This is why Chapter 6 does not simply recommend best-subset selection.
(5) Picking the winner from a huge number of models is *selection on noise*: with enough candidates, some model fits the sample well by luck, so its R^2 is optimistic and it may not generalise. That is **multiple testing** (Chapter 13), and it is also why model choice must be judged on held-out data (Chapter 5).

Common mistake

Assuming the 22 stepwise fits search the same space as the 64. Stepwise follows one *path* through the subsets, so it can miss the best model — the speed is bought with a restricted search.