Mathematical notation

Last modified: 2026-11-07 21:34:25 (PST)

> **AI-Assisted Authorship**
>
> This chapter was written with the assistance of GitHub Copilot, which expanded on outlined ideas and draft notes provided by the author. The content represents the author's perspective and has been reviewed for accuracy, but the detailed prose was generated through AI assistance.

Notation is part of your prose, and readers parse it the same way they parse a sentence. This chapter collects notational habits that make quantitative writing easier to read.

Avoiding double inequalities

Definition 0.1 (Double inequality (chained inequality))**.** A **double inequality** (or **chained inequality**) places one quantity between two others, as in $a < x < b$.

Prefer interval notation instead: $x \in (a, b)$.

Interval notation is easier to read for three reasons:

- It states one relation instead of two, so the reader parses a single claim rather than combining a pair of them.
- It names the set you actually mean, so the same expression can be reused as a domain, a constraint, or the argument of a quantifier.
- It marks inclusive and exclusive endpoints with brackets and parentheses, which are easier to distinguish at a glance than $\leq$ and $<$.

Example 0.1 (Rewriting a double inequality)**.**

✗ Suppose $0 \leq p \leq 1$ and $0 < \sigma < \infty$.

✓ Suppose $p \in [0, 1]$ and $\sigma \in (0, \infty)$.

The rewrite makes the two constraints look alike, so the reader can compare them directly instead of checking four inequality symbols.

The same guidance applies to prose: write "for $x \in (0, 1)$" rather than "for x between 0 and 1, exclusive".

When the middle term is not a single symbol

Interval notation works best when the quantity being constrained is a single expression. When the middle term is complicated, naming it first keeps the constraint short.

Example 0.2 (Naming the middle term)**.**

✗ $0 < \frac{\hat{\theta}-\theta}{\widehat{\text{se}}(\hat{\theta})} < 1.96$

✓ Let $z = \dfrac{\hat{\theta} - \theta}{\text{se}(\hat{\theta})}$; then $z \in (0, 1.96)$.

Finally, a chain that compares three *different* quantities — for example $\hat{\theta}_1 < \hat{\theta}_2 < \hat{\theta}_3$ — is not a constraint on a single quantity, so interval notation does not apply. Write such comparisons as separate statements ($\hat{\theta}_1 < \hat{\theta}_2$ and $\hat{\theta}_2 < \hat{\theta}_3$) unless the ordering of all three is the point you are making.

One operation per step

Write a derivation so that each displayed line does exactly one thing to the line before it: apply one rule, substitute one result, or simplify one piece. Give each line its own justification.

Combining steps saves the writer a line and costs every reader a calculation. A reader who cannot see how one line follows from the last has to redo the missing work on scratch paper, or take the step on trust. Either way, an error hidden inside a combined step is harder for the reader, or a reviewer, to catch. The advice on "Trivial" and "trivially" says to write out a step that takes only a line; this section applies that advice to the layout of a displayed derivation.

Example 0.1 (Splitting a combined step)**.** Suppose a derivation has reached $-2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_x x_i)$.

✗

$$-2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_x x_i) = -2(n\bar{y} - n\beta_0 - \beta_x n\bar{x})$$

✓

$$\begin{aligned}
-2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_x x_i) &= -2\left(\sum_{i=1}^{n} y_i - \sum_{i=1}^{n}\beta_0 - \sum_{i=1}^{n}\beta_x x_i\right) && \text{(distribute the sum over the terms)} \\
&= -2\left(\sum_{i=1}^{n} y_i - n\beta_0 - \sum_{i=1}^{n}\beta_x x_i\right) && \text{(sum of a constant)} \\
&= -2\left(\sum_{i=1}^{n} y_i - n\beta_0 - \beta_x\sum_{i=1}^{n} x_i\right) && \text{(factor the constant out of the sum)} \\
&= -2\left(n\bar{y} - n\beta_0 - \beta_x\sum_{i=1}^{n} x_i\right) && \left(\sum_i y_i = n\bar{y}\right) \\
&= -2(n\bar{y} - n\beta_0 - \beta_x n\bar{x}) && \left(\sum_i x_i = n\bar{x}\right)
\end{aligned}$$

The ✗ version does five operations in one line: it distributes the sum over the terms, sums a constant, factors a constant out of a sum, substitutes $n\bar{y}$, and substitutes $n\bar{x}$. The ✓ version gives each of the five its own line.

Common ways to combine steps without noticing:

- substituting a definition and then applying a rule to it (see Applying a definition is a step);
- applying the rule for the derivative of a sum and the chain rule in the same line;
- substituting a result and then simplifying it;
- multiplying constants and factoring them out of a sum;
- expanding a product and collecting like terms;
- canceling a in $a+(b-a)$ removes the parentheses $(a+b-a)$, reorders the terms $(b+a-a)$, groups the two that cancel $(b+(a-a))$, replaces $a-a$ with 0 $(b+0)$, and drops the $+0$;
- canceling n in $n \cdot \frac{b}{n}$ rewrites the division as multiplication by a reciprocal $(n \cdot (b \cdot \frac{1}{n}))$, removes the parentheses $(n \cdot b \cdot \frac{1}{n})$, reorders the factors $(b \cdot n \cdot \frac{1}{n})$, groups the two that cancel $(b \cdot (n \cdot \frac{1}{n}))$, replaces $n \cdot \frac{1}{n}$ with 1 $(b \cdot 1)$, and drops the $\cdot 1$;
- writing "setting this to zero and dividing by $-2n$ gives", which hides two operations in a sentence.

Name a step by its standard rule

Justify a line with the standard name of the rule it applies, not an informal description of what it does to the symbols. Write "distribute the sum over the terms" rather than "split the sum", and "factor the constant out of the sum" rather than "pull out the constant". A standard name tells the reader which rule to check the line against; an informal one only describes the change.

Applying a definition is a step

Replacing a named quantity with its definition is one of the most important steps in a derivation. This replacement applies the general rules to the specific problem, so never skip it or merge it into the next step. Give it its own line, justified as "definition of …". The same holds for a result from an earlier part of the same problem: substitute it on its own line, and cite the part.

Example 0.2 (Applying the definition of a vector before differentiating it)**.** Suppose the prediction error vector is defined as $\tilde{e} = \Phi\tilde{\beta} - \tilde{y}$, and a derivation needs its derivative with respect to $\tilde{\beta}$.

✗

$$\frac{\partial}{\partial\tilde{\beta}}\tilde{e} = \frac{\partial}{\partial\tilde{\beta}}(\Phi\tilde{\beta}) - \frac{\partial}{\partial\tilde{\beta}}\tilde{y} \quad \text{(derivative of a difference)}$$

✓

$$\begin{aligned}\frac{\partial}{\partial\tilde{\beta}}\tilde{e} &= \frac{\partial}{\partial\tilde{\beta}}(\Phi\tilde{\beta} - \tilde{y}) && \text{(definition of } \tilde{e}\text{)}\\ &= \frac{\partial}{\partial\tilde{\beta}}(\Phi\tilde{\beta}) - \frac{\partial}{\partial\tilde{\beta}}\tilde{y} && \text{(derivative of a difference)}\end{aligned}$$

The ✗ version applies the definition of $\tilde{e}$ and the rule for a difference in one line, and its justification names only the rule. The ✓ version shows the definition on its own line.

Using a fact the reader has not seen

When a step relies on a fact the document has not yet shown, such as deviations from a mean summing to zero, prove that fact first, in its own short derivation or exercise, and cite that proof at the step that uses the fact. Don't leave the fact as an unexplained justification.

Long derivations

Splitting every step makes derivations longer. Keep them readable by breaking them into stages: derive an intermediate result in its own block, then plug that result back into the main derivation. Naming a part of an expression that recurs in every line, such as writing d_i for $(y_i - \bar{y}) - \beta_x(x_i - \bar{x})$, can also keep each line short.

Show why you need a result before you derive it

Derive an intermediate result only after the main derivation has shown that it needs that result. For example, to differentiate a composite function:

1. Start the main derivation, and apply the chain rule. The result is a product of an inner and an outer derivative, not yet known.
2. Pause the main derivation, and derive the inner derivative in its own block.
3. Derive the outer derivative in its own block.
4. Return to the main derivation, and substitute both results.

A reader who encounters the inner derivative first does not yet know what it is for.

Color where a result is used

Give each intermediate result a text color, and use the same color wherever that result appears:

- in the line where a later step needs it, as a placeholder;
- in the line of its own derivation where it is found;
- in each line where it is substituted back in.

State the color key in words at the start of the derivation, so the derivation is still readable without color, for example in grayscale print or for readers who do not see color differences. Use `\textcolor{name}{...}`, which renders in HTML (MathJax) and in PDF.

Example 0.3 (Coloring the two factors of a chain rule)**.** Writing the mean squared error as $\frac{1}{n}\tilde{e} \cdot \tilde{e}$, with $\tilde{e} = \Phi\tilde{\beta} - \tilde{y}$, the chain rule gives

$$\frac{\partial}{\partial \tilde{\beta}} \mathrm{MSE}(\tilde{\beta}) = \left(\frac{\partial}{\partial \tilde{\beta}} \tilde{e}\right) \left(\frac{\partial}{\partial \tilde{e}} \frac{1}{n} \tilde{e} \cdot \tilde{e}\right).$$

Separate blocks then find $\frac{\partial}{\partial \tilde{\beta}} \tilde{e} = \Phi^\top$ and $\frac{\partial}{\partial \tilde{e}} \frac{1}{n} \tilde{e} \cdot \tilde{e} = \frac{2}{n} \tilde{e}$, and the main derivation resumes with $\Phi^\top \left(\frac{2}{n} \tilde{e}\right)$. The colors show at a glance which block each factor came from.

One equals sign per line

Put at most one equals sign on each line of math. When a result takes two or more equalities, write them as an aligned display, with one equality per row, each row starting with `&=`.

A chain such as $a = b = c$ on one line asks the reader to find where each step starts and ends, and to check every step at once. An aligned display puts each step on its own row, so the reader can check one row at a time, and the left-hand side, written once, stays visible above the rows that rewrite it. The rows also leave room for a justification at the end of each step, as One operation per step asks.

Example 0.1 (Splitting a chain into rows)**.**

✗

$$\hat{B} = \arg\min_B \|Y - XB\|_F^2 = \arg\min_B \sum_{k=1}^{C} \|Y_{*,k} - XB_{*,k}\|^2$$

✓

$$\begin{aligned} \hat{B} &= \arg\min_B \|Y - XB\|_F^2 \\ &= \arg\min_B \sum_{k=1}^{C} \|Y_{*,k} - XB_{*,k}\|^2 \end{aligned}$$

The ✗ version makes the reader find the second equals sign in a long line before they can compare the two expressions. The ✓ version sets the two expressions one above the other, starting at the same place.

The rule applies wherever an equals sign appears:

- **Inside an aligned display.** A row such as `&= b = c` is still a chain; split it into two rows.
- **In running text.** An inline chain such as ✗ $\bar{x} = \frac{6}{3} = 2$ becomes an aligned display, or two statements in the sentence.
- **For a statement that several quantities are equal.** Write ✓ $H_0 \colon \beta_j = 0$ for every $j \in \{1, \dots, p\}$ rather than ✗ $H_0 \colon \beta_1 = \beta_2 = \cdots = \beta_p = 0$, so that each claim is a single equality.

Separate equations set side by side, such as $x = 1, \quad y = 2$, each have one equals sign, so they are not chains. They still read more easily as separate rows, or as separate statements in the text.

Functions and their values

A function and its value at a point are different objects. In $\mu_i = \mu(x_i)$, μ is a function, and $\mu(x_i)$ is a number: the value of μ at x_i. Written with its placeholder argument, $\mu(x)$ already denotes a value, so "the value of $\mu(x)$ at x_i" names a value of a value. Either drop the placeholder and write "the value of μ at x_i", or keep the placeholder and name the substitution: "$\mu(x)$ evaluated at x_i".

Example 0.1 (Saying where a function is evaluated)**.**

✗ Each outcome is centered on the value of a mean function $\mu(x)$ at its own covariate values.

✓ Each outcome is centered on a mean function $\mu(x)$ evaluated at that outcome's covariate values.

✗ the value of the density $f(x)$ at 0

✓ the value of f at 0, or $f(0)$

✗ the likelihood $L(\theta)$ at $\hat{\theta}$

✓ $L(\theta)$ evaluated at $\hat{\theta}$, or $L(\hat{\theta})$

In each ✗ version, "the value of" or "at" sits next to a placeholder argument, so the reader has to decide whether $\mu(x)$, $f(x)$ or $L(\theta)$ means the function or one of its values. The first ✗ version also has an ambiguous "its": the nearest candidate is the mean function, not the outcome.

Use the notation to match:

- Write μ, or "the mean function", for the function itself.
- Write $\mu(x)$ with a placeholder argument where you introduce or define the function, to show what its argument is, as in "a mean function $\mu(x)$" or $\mu(x) := \beta_0 + \beta_x x$, and say "evaluated at" when you substitute a point into it.
- Write $\mu(x_i)$, or "the value of μ at x_i", for a value.

Definition 0.1 (Fitted value)**.** The **fitted value** of observation i is the value that a fitted model predicts at that observation's input. In machine learning notes, where the fitted prediction function is h, it is $\hat{y}_i = h(x_i)$.

Define the fitted value once, then print $\hat{y}_i$, not $h(x_i)$, wherever it appears in a loss, a residual or a risk sum. Keep $h(x)$ where h itself is the subject, as in "the linear function $h(x) = \beta_0 + \beta_x x$", and where the output is not a prediction of y_i, such as the decision score of a support vector machine or a vector of class probabilities.

In the source, a document that uses the shared macros writes the fitted value with the macro for a predicted outcome, `\eoutvar` (see Using shared macros), and a document without them may write `\hat{y}`.

Types of functions and procedures

Name the type of every function and procedure when you define it:

- the set it takes its argument from (its domain)
- the set its values lie in (its codomain)
- for each input and output, what kind of mathematical object it is

Say "function", not "rule" or "map".

A procedure that takes several inputs is a function of all of them, so list each input with its type. When an output is itself a function, say so and give its domain and codomain too.

A function's argument is not part of what specifies the function. Give the argument in the definition, as the domain, and list only the other inputs as what specifies the function.

Example 0.1 (Typing a procedure's inputs and output)**.**

> ✗ A fitting procedure is a rule that takes data and produces a prediction.
>
> ✓ A fitting procedure is a function from data sets to fitted models. A fitted model is a function from covariate values to predicted outcomes.

The ✗ version leaves four types unsaid. "Rule" does not say that the procedure is a function. "Data" does not say that the input is a data set. "A prediction" is a single predicted outcome, so it does not say that the output is a fitted model. Nor does it say that a fitted model is itself a function, which makes a prediction only when given a covariate value. The ✓ version states each type, so a reader can check that the output of one is an acceptable input to the next.

> ✗ A fitting procedure is specified by the data set it is fitted to, its model family and its objective function.
>
> ✓ A fitting procedure is specified by its model family and its objective function. It takes a data set as its argument.

The data set is the argument of the fitting procedure, so it is not one of the inputs that specify the procedure.

Check each use against the stated types: a function applied to something outside its domain, or a value described with the wrong type, is an error even when the notation looks right.

Random variables and their values

A random variable and a value that it takes are different objects. Before the data are collected, the outcome is a random variable; after, it is a number. Statements about the two have different meanings. Write them with different symbols.

Use a capital letter for the random variable and the matching lowercase letter for a value that it takes:

- Y is the random variable, such as the outcome of a coin flip before the flip;
- y is one value of Y, such as 1 after the flip lands heads;
- Y_i is the random variable for the i-th observation;
- y_i is that observation's observed value.

State the convention where you first use the symbols. Keep it in every later section. Do not switch between Y_i and y_i for the same quantity unless the meaning changes from "before the data" to "after the data".

Example 0.1 (Writing a random variable and its value)**.**

✗ The probability that y_i exceeds 3 is 0.2.

✓ The probability that Y_i exceeds 3 is 0.2.

✗ The sample mean of the Y_i is 2.4.

✓ The sample mean of the y_i is 2.4.

A probability is a statement about a random variable. The ✗ version of the first pair puts a fixed number where a quantity that can vary belongs. An observed number has no probability of exceeding 3. The second pair has the reverse mistake. A sample mean of 2.4 is a number computed from observed values. The ✗ version attributes it to the random variables. The sample mean of the random variables, $\bar{Y}$, is itself a random variable.

Match the notation to the claim:

- Write $\Pr(Y \leq y)$ for a probability, with the random variable on the left of the inequality and the value on the right.
- Write $\mathrm{E}[Y]$ and $\mathrm{Var}(Y)$ for the mean and variance of a random variable, never $\mathrm{E}[y]$ or $\mathrm{Var}(y)$ for an observed value.
- Write a statistic computed from observed values with lowercase letters, as in $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, and the same statistic as a random variable with capitals, $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$. The first is a number, such as 2.4. The second has a sampling distribution.

Name the kind of object in the words, too, not only in the symbol. Call a random variable a "variable", such as "the output variable Y", and call a number a "value", such as "the input value x" or "the observed value y". Do not say "fix x" or "let x be fixed". It does not say whether x is a variable held constant or a value you chose. Say "choose an input value x", or "take the observed value y as given".

Example 0.2 (Naming a variable and its value in the words)**.**

✗ Fix an input x, and let $Y = \mu(x) + \varepsilon$.

✓ Choose an input value x, and let the output variable $Y = \mu(x) + \varepsilon$.

The ✗ version leaves the reader to work out that x is a number you choose and Y is random. The ✓ version names the input as a value and the output as a variable, which is what the symbols say.

Some fields reserve capital letters for matrices, or use the same letter for a random variable and its value. In that case, say so where you introduce the symbols. Add a word that tells the reader which meaning applies, such as "the random variable Y_i" or "the observed value y_i".

Matching names and notation

A name and its notation must describe the same object. When you name a quantity in words and also show it in symbols, use the symbols that go with that name. A reader who sees the name "dot product" expects to see the notation for a dot product. If the symbols look like a different operation, the reader has to work out whether the name or the symbols is wrong.

Keep the pairing the same everywhere the quantity appears. Notes, assignments, solutions and slides in one course should use one name and one notation for each quantity. When two fields call one idea by different names, choose one name, list the other names where you define the idea, and use only the chosen name afterward (see Name the synonyms).

Example 0.1 (Matching a name to its notation)**.**

✗ Compute the dot product $\tilde{a}'\tilde{b}$.

✓ Compute the dot product $\tilde{a} \cdot \tilde{b}$.

The ✗ version names a dot product but writes a matrix product with a transpose. The two are equal for column vectors, but the reader sees two names for one quantity. If you want to show that they are equal, say so in a sentence, as in "the dot product $\tilde{a} \cdot \tilde{b}$ equals the matrix product $\tilde{a}'\tilde{b}$". Writing dot products says when to use each form.
In a course that writes regression in statistical notation:

✗ Consider the model $h(x) = wx + b$.

✓ Consider the mean function $\mu(x) = \beta_0 + \beta_x x$.

The ✗ version names the same quantity as the rest of the course with a different word and different symbols. A student then has to learn that h, w and b are μ, β_x and β_0.

Before you release a document that reuses a quantity from another one, search both for its name and its symbols, and make them agree.

Label every inline symbol

In prose, write a text label next to every inline math symbol. The label tells the reader what the symbol stands for without making them search for its definition. Put the label before the symbol, as in "the population noise variance, σ^2".

Example 0.2 (Labeling a symbol in a list of quantities)**.**

✗ Which of the three quantities (squared bias, variance of $\hat{y}$, σ^2) can a better prediction method reduce?

✓ Which of the three quantities (the squared bias, the variance of the prediction $\hat{y}$, and the population noise variance σ^2) can a better prediction method reduce?

In the ✗ version, the reader sees σ^2 with no word for it. In the ✓ version, each quantity has a label, and the symbol that follows the label confirms which quantity is meant.

Using shared macros

Write each symbol through a shared macro file, and write the macro everywhere the symbol appears. A group that writes the same quantity in raw LaTeX in one document and with a macro in another ends up with two notations for one quantity. Changing a convention then means editing every document by hand.

Before you write any symbol, search the group's shared macro file for the concept it names. If a macro names the concept, use it. If none does, add one to the shared file before you use the symbol, so the next document finds it.

Example 0.1 (Using a shared macro)**.** This example compares the source you type, so both versions appear as code.

✗ Source: `Let $\beta_1$ be the slope and write the sum as $\sum_{i=1}^n$.`

✓ Source: `Let $\coef{1}$ be the slope and write the sum as $\sumin$.`

The ✗ version writes the slope and the sum in raw LaTeX. The ✓ version names each concept once, in the shared file. When the group changes how it prints a coefficient or a sum, every document that uses the macro changes with it.

The same rule covers these quantities:

- transposes
- hats on estimates
- expectations
- variances
- probabilities
- vectors

A quantity that has a macro keeps the same symbol in every document.

Name a macro for what the symbol means, not for the letter it prints. A macro such as `\vdelta` or `\hbeta` only spells a Greek letter, so it names no concept, and changing the symbol for that concept still means editing every use. Outside a passage that discusses the notation itself, do not write a Greek letter in a math expression, either as a raw command such as `\delta` or inside a letter-named macro such as `\vdelta`. The only exception is a one-off symbol that fits no semantic category, and the rule for Latin letters below says when that applies.

Example 0.2 (Naming a macro for its meaning)**.**

✗ Source: `The error at layer $\ell$ is $\vdelta^{(\ell)}$, and the step size is $\eta$.`

✓ Source: `The error at layer $\ell$ is $\backerr^{(\ell)}$, and the step size is $\learnrate$.`

Both versions print the same symbols. In the ✓ version, a reader of the source sees what each symbol stands for, and a group that changes the symbol for the step size changes one line in the shared file.

A passage about the notation itself, such as a table that lists each symbol and what it means, may write the letters directly.

The same reasoning applies to decorators. A hat, a tilde or a bar says how a symbol is drawn, not what it means. A hat can mark an estimate, a fitted value or a Fourier transform, so `\hat`

names no concept either. Write the macro that names the role instead, such as `\est{}` for an estimate, and keep `\hat` for passages that discuss the notation itself. New math follows this now; existing hats are converted in separate sweeps.

Example 0.3 (Naming a decorator for its meaning)**.**

✗ Source: `The estimate $\hat{\mean}$ is close to $\mean$.`

✓ Source: `The estimate $\est{\mean}$ is close to $\mean$.`

Both versions print the same hat. If the group later marks estimates some other way, only the definition of `\est` changes.

Font commands follow the same rule. `\mathscr`, `\mathcal`, `\mathbb` and `\mathbf` say how a letter is drawn, not what it stands for, so a raw `\mathbb{R}` or `\mathscr{A}` leaves the reader of the source guessing at the concept and leaves the group editing every use to change the symbol. Write the macro that names the concept. Name the macro for the set, such as `\reals` for the real numbers, not for the font, as `\bbR` would be. A document that reaches for a font command directly has found a concept with no macro yet: add the macro to the shared file, then use it. As with Greek letters, a passage that discusses the notation itself may write the font command directly.

Example 0.4 (Naming a set for its meaning)**.**

✗ Source: `The function maps $\mathbb{R}$ to $\mathbb{R}$.`

✓ Source: `The function maps $\reals$ to $\reals$.`

Both versions print the same symbol. If the group changes how it writes the real numbers, only the definition of `\reals` changes.

Latin letters follow the same rule, with the same single exception for letters of either alphabet. Outside a passage that discusses the notation itself, do not write a literal Latin or Greek letter in a math expression. The exception is a one-off symbol that fits no semantic category. The shared macros already name outcome variables, covariates and generic parameters, so a one-off should almost never be needed. When a symbol has a category but no macro, add the macro to the shared file first. New macros always go in the group's shared macros repository, never in a repository that applies them, unless that repository explicitly overrides the default meaning of one specific macro.

Latin letters with a fixed role show the rule. In a regression or prediction model, use these macros:

- `\outvar` for the outcome variable, instead of a bare `y`;
- `\eoutvar` for a predicted outcome;

- `\predvar` for a predictor, instead of a bare `x`;
- `\resid` for a residual, the observed minus the predicted outcome;
- `\prederr` for a prediction error, the predicted minus the observed outcome.

For an observation used to fit the model, the residual is the negative of the prediction error. Write `\resid` for the observed minus the predicted outcome and `\prederr` for the predicted minus the observed outcome, never one for the other. For an estimate of a parameter, `\erf{\eparam}` is the estimation error, the estimate minus the true value; it prints ε, the symbol the shared macros use for an error. These macros need a version of the shared macros that includes them, so update a repository's copy before using them there. The outcome, predictor and residual macros also have random-variable and vector forms, such as `\Outvar`, `\voutvar` and `\vresid`.

Example 0.5 (Naming a variable for its role)**.**

✗ Source: `The residual is $e_i = y_i - \hat{y}_i$.`

✓ Source: `The residual is $\resid_i = \outvar_i - \eoutvar_i$.`

The ✓ version prints the residual as r, the symbol the shared macros use for it; the other letters are the same in both. In the ✓ version, a reader of the source can tell which letter is the outcome and which is the residual.

Once a quantity has a name and a macro, write the macro instead of the expression that defines it. Each equation then stays short, and a reader sees which quantity it is about.

Example 0.6 (Writing a named quantity by its name)**.**

✗ Source: `The gradient of the mean squared error is $\frac{2}{n} \tp{\design} \paren{\design\vcoef - \vy}$.`

✓ Source: `The gradient of the mean squared error is $\frac{2}{n} \tp{\design} \vprederr$.`

The macro `\vprederr` prints the vector of prediction errors. The document defines it once, where the model is introduced, as `\vprederr = \design\vcoef - \vy`. The ✓ version has one fewer level of brackets, and it names the quantity the gradient depends on.

Delimiters follow the same rule. Manual sizing commands, such as `\big` and `\left` with `\right`, say how a bracket is drawn, not what it groups. The shared macros `\paren{}`, `\sb{}`, `\cb{}`, `\set{}` and `\abs{}` size themselves to their contents, so write them instead of a hand-sized pair. Arrows such as `\leftarrow` are not delimiters.

Example 0.7 (Letting the macro size the bracket)**.**

✗ Source: `The mean is $\E{\left( X - \mu \right)^2}$.`

✓ Source: `The mean is $\E{\paren{X - \mu}^2}$.`

Both versions print the same expression. In the ✓ version, the macro sizes the bracket, and a group that changes how brackets are drawn edits one definition in the shared file.

Macros are for the author, not the reader. Reader-facing text, such as a lecture, a handout or a notation section, describes the symbols that a macro prints and what they mean, and never names the macro that produced them. List every symbol that the reader encounters in the document's notation table, with its meaning.

Example 0.8 (Describing the symbol, not the macro)**.**

✗ Source: `The macro \avgfitf prints the average fit, so write $\avgfitf{x}$ for it.`

✓ Source: `The average fit at $x$ is written $\avgfitf{x}$.`

The ✓ version tells the reader what the symbol means. The reader never sees the macro name, and the symbol has its own row in the notation table.

Making expressions concise

Make each expression as concise as possible. The less clutter an expression has, the easier it is to read. This is one case of building every unit from smaller atomic units.

Before you finish a definition, theorem or proof, scan it for a sub-expression that is written out in full more than once, or that is a conceptually coherent chunk even if it appears only once, such as the prediction error (the fitted value minus the outcome) or the loss of a prediction on one observation. Give the chunk a name and a macro, and write the macro everywhere the chunk appears.

Example 0.1 (Naming a repeated chunk)**.**

✗ Source: `The loss is $\frac{1}{n} \sum_{i=1}^n \paren{\eoutvar_i - \outvar_i}^2$, and its derivative is $\frac{2}{n} \sum_{i=1}^n \paren{\eoutvar_i - \outvar_i} \predvar_i$.`

✓ Source: `The loss is $\frac{1}{n} \sum_{i=1}^n \prederr_i^2$, and its derivative is $\frac{2}{n} \sum_{i=1}^n \prederr_i`

```
\predvar_i$.
```

Both versions print the same quantities. The ✗ version writes the difference `\eoutvar_i - \outvar_i` out twice. The ✓ version names it once, as the prediction error, and each formula shows at a glance what it is built from.

Write each expression in its most compact form, built from named sub-pieces that are defined separately. Use one form per expression, and keep it simple. For example, write the expected squared prediction error as the squared bias plus the variance plus the residual variance, and define each of the three pieces on its own, instead of expanding them inside the formula.

Search the shared macros for the chunk before you write it by hand. A macro often exists already, and any expression spelled out in full that a macro or a shorter form could say is the sign that nobody looked. A difference is only the commonest case. For example, an exercise wrote the left side of the bias–variance decomposition as `\E{\paren{\newoutvar - \predvalf{\newpredval}}^2}`, when `\prederrf{\predvalf{\newpredval}}` names the same prediction error, and the squared bias, written out the same way, has `\biassymf`. When you find one such chunk, sweep the other notes repositories for it, in small batches of one repository each.

Add the macro to the shared macro file before you use it, never as a local macro in one document (see Using shared macros), and record its meaning in that file's `interpretations.tsv`.

Abstracting sub-expressions you are not manipulating

Replace each sub-expression that you are not manipulating with a symbol that stands for it. Keep a sub-expression written out only on the line where you are about to apply its definition, a rule or a theorem to it. Every other line shows the symbol.

This is one case of making each expression as concise as possible. It also asks for the habit behind that advice: split a long expression into short named pieces, and give each piece the symbol or shared macro that names its concept.

Choose the symbol to match the thing it names. For the population mean $\mathbb{E}[X]$, write μ. Add a subscript, such as μ_X, only when the context holds a second mean and the reader could confuse the two. State the abstraction once, in a sentence such as "Let $\mu = \mathbb{E}[X]$", before the first line that uses the symbol.

Example 0.1 (Abstracting the population mean)**.** Suppose a derivation expands the variance of a random variable X.

✗

$$\begin{aligned}\mathbb{E}[(X-\mathbb{E}[X])^2] &= \mathbb{E}\left[X^2 - 2X\mathbb{E}[X] + \mathbb{E}[X]^2\right] \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2\end{aligned}$$

✓ Let $\mu = \mathbb{E}[X]$.

$$\begin{aligned}\mathbb{E}[(X-\mu)^2] &= \mathbb{E}[X^2 - 2\mu X + \mu^2] && \text{(expand the square)} \\ &= \mathbb{E}[X^2] - 2\mu\mathbb{E}[X] + \mu^2 && \text{(linearity of expectation)} \\ &= \mathbb{E}[X^2] - 2\mu\mu + \mu^2 && \text{(definition of } \mu\text{)} \\ &= \mathbb{E}[X^2] - \mu^2 && \text{(combine like terms)}\end{aligned}$$

In the ✗ version, $\mathbb{E}[X]$ is a constant that no step manipulates until the last one, yet it clutters every line. In the ✓ version, only the line that uses the definition of μ needs to know what μ stands for.

Writing dot products

When two vectors multiply to give a number, write the product as a dot product, $\tilde{x} \cdot \tilde{\beta}$. Do not write it as a transpose product, $\tilde{x}'\tilde{\beta}$, or with inner-product brackets, $\langle \tilde{x}, \tilde{\beta} \rangle$. The dot shows one operation on two vectors. A transpose product asks the reader to picture a row vector times a column vector and to work out that the result is a single number.

Example 0.1 (Writing a dot product)**.**

✗ The model is $f(\tilde{x}) = \tilde{x}'\tilde{\beta}$.

✓ The model is $f(\tilde{x}) = \tilde{x} \cdot \tilde{\beta}$.

Both versions give the same number. The ✓ version shows it as one product of two vectors. In a group that uses shared macros (see Using shared macros), write the ✓ version with the dot-product macro, for example `$\dprod{\vx}{\vbeta}$`.

Keep the transpose where the product is not a dot product of two vectors:

- a product of matrices, such as $X'X$
- an outer product, such as $\tilde{x}\tilde{x}'$
- a quadratic form, such as $\tilde{x}'A\tilde{x}$

Keep the inner-product brackets where the text means any inner product, such as a definition of an inner product space.

Using the notation that best represents the concept

Use the notation that best represents the concept being depicted. Choose the symbol, the macro and the name by asking what the object is, not by asking what is quicker to type or what the nearby text used. A symbol that fits a different concept sends the reader to the wrong idea.

The rules that follow are cases of this principle:

- The predictor matrix and the design matrix are different objects, so they get different symbols.
- The mean function is a mean, so it is μ, not f.
- The fitted prediction function is a prediction function, so it is named as one, not as a fitted mean function.
- A reference formula is a general statement, so it is written in its general form, not for one example.

When no rule covers a case, ask the same question: which symbol, name and form best represents the concept here? Then record the answer as a rule, with this principle as its reason.

Writing the predictor and design matrices

Write a matrix in matrix styling, not in the plain italic of a scalar. This holds for the predictor matrix $\mathbf{X}$ in every file you edit: notes, assignments, solutions, slides and note sheets. A plain italic X suggests a random variable or a single number. In a group that uses shared macros (see Using shared macros), write the predictor matrix with the matrix macro `\mX`.

Keep two objects apart, because they are not the same matrix:

- The *predictor matrix* holds the raw inputs. Its rows are the predictor vectors $\underset{\sim}{x}_i$ of the n observations. Write it `\mX`.
- The *design matrix* holds the inputs after any feature map and the intercept trick. Its rows are the design vectors $\underset{\sim}{D}(\underset{\sim}{x}_i)$. Write it `\design(\mX)`, or `\design` for short.

The same split holds for one observation: the predictor vector is `\vx`, and the design vector is `\vfeatmap(\vx)`. Without a feature map or an intercept column, the design matrix equals the predictor matrix, but say so rather than reuse one symbol for both.

Example 0.1 (Predictor matrix and design matrix)**.**

✗ Let X be the design matrix, and fit $\hat{\beta} = (X'X)^{-1}X'y$.

✓ Let $\mathbf{X}$ be the predictor matrix and $D(\mathbf{X})$ the design matrix. The fitted

coefficients solve the normal equations for $D(\mathbf{X})$.

The ✓ version shows which matrix holds the raw inputs and which holds the columns the model uses.

Stating formulas in their general form

On a reference sheet, a formula sheet or a summary, state each formula in its general form. Do not state it only for a specific example. A reader looks a formula up to apply it to a new case, and an example shows only one case.

Example 0.1 (Stating the chain rule)**.**

✗ $\frac{d(u^2)}{dw} = 2u\frac{du}{dw}$

✓ $\frac{df(u)}{dw} = f'(u)\frac{du}{dw}$

The ✗ line is the chain rule for squaring only. The ✓ line holds for every differentiable f, and the squaring case follows by taking $f(u) = u^2$.

Put a specific case on the sheet only as a labeled example next to the general formula.

Writing the mean function and the prediction function

Write the mean function as μ, not as f. The mean function is the population mean of the outcome given the inputs, $\mu(x)$. In a group that uses shared macros (see Using shared macros), write it `\truemeanf{x}`.

Name the object you mean. When the text refers to the fitted prediction function as a concept, the rule that turns an input into a predicted outcome, write the fitted prediction function, not the fitted mean function. Write it `\epredfnf{x}` (symbol $\widehat{f}$) for an estimated one, or `\predfnf{x}` (symbol f) for any prediction function. Use the fitted or estimated mean function only where the text is about estimating the mean function μ itself.

Example 0.1 (Mean function and fitted prediction function)**.**

✗ The expected squared error of the fitted mean function at x is $\mathbb{E}\left[(y - \widehat{f}(x))^2\right]$.

✓ The expected squared error of the fitted prediction function at x is $\mathbb{E}\left[\left(y - \widehat{f}(x)\right)^2\right]$, and its bias is $\mathbb{E}\left[\widehat{f}(x)\right] - \mu(x)$.

The ✓ version names the fitted prediction function where the concept is a prediction, and keeps μ for the mean it is compared with.

Writing the log-odds

Use η (the Greek letter eta) for the log-odds, not z or another letter. In a group that uses shared macros (see Using shared macros), write it `\logodds`. Use it wherever the quantity is the log-odds, including the argument of the logistic function, so that the same quantity has one symbol in every chapter and figure. Label a figure axis that shows it "Log-odds, η" (see Scientific figures).

Example 0.1 (Argument of the logistic function)**.**

✗ The logistic function is $\sigma(z) = 1/(1 + e^{-z})$, evaluated at the log-odds z.

✓ The logistic function is $\text{expit}(\eta) \stackrel{\text{def}}{=} \frac{1}{1+e^{-\eta}}$, evaluated at the log-odds η.

The ✓ version uses the lab's symbol for the log-odds, so a reader who has seen η elsewhere recognizes it here.

Writing out notational shorthands

A notational shorthand drops part of an expression that the writer expects the reader to fill in. The commonest case is leaving the limits off a sum, product, or integral, as in $\sum_x f(x)$. In permanent writing (notes, papers, and published slides), write the full form instead: $\sum_{x \in \mathcal{R}(X)} f(x)$, where $\mathcal{R}(X)$ is the set of values that X can take.

The full form costs the writer a few characters and saves every reader a guess. A reader who sees $\sum_x$ has to work out which values of x the sum runs over, and readers who work out different answers will disagree about what the expression means. The full form also exposes mistakes. If a count variable takes only non-negative values, a sum written as $\sum_{x \in \mathbb{Z}}$ shows the wrong set at a glance, while $\sum_x$ hides the same mistake.

Give the limits of every sum, product, and integral, and state the set that any other index runs over. Board work during a lecture is often less complete, but it should still aim for the full form. The notation page of the *Math for Data Science* notes catalogs common shorthands and their full forms.

Example 0.1 (Writing out the limits of a sum and the range of an index)**.** For a discrete random variable X:

✗ $\mathrm{E}[X] = \sum_{x} x \Pr(X = x)$
✓ $\mathrm{E}[X] = \sum_{x \in \mathcal{R}(X)} x \Pr(X = x)$

For the likelihood of n independent observations $x_1, \dots, x_n$:

✗ $L(\theta) = \prod_{i} f(x_i; \theta)$
✓ $L(\theta) = \prod_{i=1}^{n} f(x_i; \theta)$

Each full form tells the reader the set the operator runs over: the values X can take, and the indices of the n observations.

Stating assumptions before using them

State each assumption before the step that uses it. A reader processes a derivation line by line, and a condition that arrives after the line it governs forces them to go back and reread that line with the condition in mind. A first-time reader may not realize the line was incomplete, and may conclude that the step is wrong or unjustified.

This covers every kind of condition:

- a restriction on a variable ("for $x \neq 2$");
- a hypothesis of a theorem;
- a case split;
- a fact that a step relies on.

Introduce the variable and its constraint together, as in "let $x \neq 2$", then write the steps that need the constraint. When a step uses a fact proved earlier, cite it in the justification on that line.

An assumptions list and a notation list are different lists:

- An assumptions list states conditions that the result depends on. Write it as a bullet list under "Assume:".
- A notation list names objects. Write it as a bullet list under "Let:".

A condition that can fail is an assumption. A parametrization is not: it writes a quantity in terms of named parameters or pieces, so it holds by definition and goes under "Let:". Do not label a parametrization as an assumption, because that tells the reader a condition is being imposed when none is.

Write the lead-in word once, before the list, and make each bullet complete the lead-in sentence. Do not repeat the word in every bullet, and do not mix the two kinds of list in one list. When both are needed, write the notation list first, because the assumptions use the names it introduces.

Example 0.1 (Stating the restriction on x before using it)**.** To show that $x = 2$ is a strict local minimizer of $f(x) = (x-2)^2$, we show that $f(x) > f(2)$ for every x near 2 other than 2 itself.

✗

$$\begin{aligned} f(x) - f(2) &= (x-2)^2 - 0 \\ &> 0 \qquad\qquad (\text{for every } x \neq 2) \end{aligned}$$

✓ Let $x \neq 2$. Then

$$\begin{aligned} f(x) - f(2) &= (x-2)^2 - 0 \quad (\text{definition of } f) \\ &> 0 \qquad\qquad (x - 2 \neq 0, \text{ so its square is positive}) \end{aligned}$$

In the ✗ version, the second line is false for $x = 2$, and the reader learns this only from the note at its end. In the ✓ version, "let $x \neq 2$" comes first, so the reader reads each line already knowing which values of x it covers, and the justification says exactly where the condition is used.

Example 0.2 (Writing several assumptions as one list)**.**

✗ Assume the sample size n is larger than 30. Assume the observations are independent. Assume the population variance σ^2 is finite.

✓ Assume:

- the sample size n is larger than 30,
- the observations are independent, and
- the population variance σ^2 is finite.

The ✓ version writes "Assume" once, and the bullets show that the three assumptions are parallel and separate.

Example 0.3 (Writing several names as one list)**.**

✗ Let n be the sample size. Let $\bar{y}$ be the sample mean. Let s be the sample standard deviation.

✓ Let:

- n be the sample size,
- $\bar{y}$ be the sample mean, and
- s be the sample standard deviation.

These bullets name objects and assume nothing about them, so this is a notation list. If the result also needed a condition such as a sample size n larger than 30, a separate assumptions list after this one would state it.

Example 0.4 (Defining the noise is a parametrization, not an assumption)**.**

✗ Assume:

- the outcome variable is $Y = \mu(x) + \varepsilon$, and
- the noise variable ε has population mean 0.

✓ Let:

- $\mu(x)$ be the population mean of the outcome variable Y given the input value x, and
- $\varepsilon = Y - \mu(x)$ be the noise variable.

Then $Y = \mu(x) + \varepsilon$, and the population mean of ε given x is 0.

In the ✗ version, the reader is told that the decomposition and the zero mean are conditions on the data. They are not. Defining $\varepsilon = Y - \mu(x)$ makes both statements true for every distribution of Y that has a finite population mean. The ✓ version puts the definitions under "Let:" and derives the zero mean. A real assumption, such as the noise variable being independent of the input value, can fail, so it belongs under "Assume:".