# How Disability Weights Are Estimated: A Guide to the WelfareWeights Engine

Richard Bruns
Johns Hopkins University
bruns@jhu.edu • ORCID 0000-0002-2209-4242

The WelfareWeights Project: welfareweights.com • github.com/welfareweights/engine

July 2026

This is the auditing document for the WelfareWeights engine, an open implementation of the estimation pipeline behind the Global Burden of Disease disability weights (Salomon et al. 2012). It explains what the estimator does and why, at the level of someone who took econometrics in grad school years ago, or someone fluent in math and code who never took it. Appendix A carries the referee-facing validation evidence; every number in it is reproducible from the repository's committed tests and studies with recorded seeds. Code objects throughout are live links into the repository.

**Notation.** $\Phi$ is the standard normal CDF; $\text{logit}(p) = \log \frac{p}{1-p}$ maps probabilities $(0, 1)$ onto the real line; expit is its inverse. "MLE" is maximum likelihood estimation: pick the parameter values under which the observed data was most probable.

## 1 What a disability weight is, and what it is used for

A disability weight is a number between 0 and 1 that says how bad a year lived with a health state is: 0 is full health, 1 is as bad as being dead. Burden-of-disease accounting multiplies time lived with a condition by its weight, so every DALY figure you have ever seen has a table of these weights underneath it.

Concretely: with a weight of 0.30, ten years lived with the condition contribute $0.30 \times 10 = 3.0$ DALYs (years of healthy life lost to disability); a death at 40 with life expectancy 80 contributes 40 DALYs. A program that cures 1,000 people of a 0.30-weight condition for 10 years averts 3,000 DALYs; at a cost of $6 million that is $2,000 per DALY averted, and that number is what gets compared across programs. The weights carry real decisions: cut the 0.30 to 0.20 (a plausible disagreement between elicitation methods) and the program's cost-effectiveness worsens by a third, easily enough to move it across a funding threshold. This sensitivity is why it matters that the weights be reproducible and auditable, and why this engine exists.

The formal rulebooks for this arithmetic are the reference-case guidelines for benefit-cost analysis in global health and development (Robinson et al. 2019, which cites Salomon et al. 2012 by name as the source of the weights it monetizes) and, for US rulemaking, the HHS Guidelines for

Regulatory Impact Analysis. Both take health-state weights as inputs to the benefit calculation; producing those inputs transparently is exactly the step this engine makes auditable.

And these are not academic abstractions. GiveWell's cost-effectiveness models multiply GBD disability weights directly into the value of every recommended charity; where GiveWell distrusts a published weight it overrides it and says so (schistosomiasis, where it uses 0.05 against the GBD's 0.005–0.006), which is precisely the kind of scrutiny transparent weights invite. Open Philanthropy's global health funding bar is roughly $50 per DALY averted, with the weights in the denominator of every grant screened against it. Pakistan's national essential health services package was assembled by ranking 117 interventions on cost-per-DALY figures built on these weights. Gavi's board prioritizes new vaccine investments partly on procurement cost per DALY averted.

The weights come from surveys, and the surveys deliberately do NOT ask "rate this condition from 0 to 1" (people are bad at that). They ask two kinds of easier questions, and the estimation problem is to reassemble the 0-to-1 scale from the answers. The engine implements the reassembly used for the GBD weights (Salomon et al. 2012, 2015), in three stages.

## 2 The two question types

**Paired comparison (PC).** The respondent reads lay descriptions of two health states and answers one thing (template: `pc_question_text` in `src/welfareweights/survey.py`):

> Two people have the same life expectancy. Assume each condition lasts for the rest of their life.
> Person A [description of state 1]
> Person B [description of state 2]
> Who is healthier overall, A or B?

Each respondent answers a handful of these with randomly assigned pairs. PC answers are informative about the *order and spacing* of states relative to each other, but they contain no information about where the whole scale sits; every answer would be identical if all states were twice as bad, or uniformly milder.

**Population health equivalence (PHE).** The question that pins the scale (template: `phe_question_text`):

> Two health programs cost the same. Which produces the greater health benefit?
> Program 1 prevents 1000 people from dying.
> Program 2 prevents [$c$] people from having [state $s$] for the rest of their life.
> Which program, 1 or 2?

Here $c$ varies (the GBD instrument uses 1500, 2000, 3000, 5000, 10000). If you value averting one death at 1 and averting a lifetime case of state $s$ at its weight $DW_s$, program 1 is better exactly when $1000 > c \cdot DW_s$, that is, when $DW_s < 1000/c$. So each answer reveals which side of a known threshold the respondent's valuation falls on; and because the trade is against deaths, the threshold is expressed on the 0-to-1 scale we ultimately want, with death $= 1$ built in.

One convention to swallow before going further: reading the choice as "1000 vs. $c \cdot DW_s$" treats averting a death and averting a lifetime condition as covering the same person-time. That is the DALY convention of the instrument this replicates; it is inherited here, not introduced here.

## 3 Stage A: paired comparisons to a relative scale

*Code: `fit_pc` in `src/welfareweights/probit.py`; design matrix in `src/welfareweights/design.py`.*

**The model.** Give every state $i$ a latent "healthiness" number $\theta_i$. When a respondent compares states $i$ and $j$, they do not perceive $\theta$ exactly; they perceive it with noise, $H_i \sim N(\theta_i, \sigma^2)$ independently for each state in the pair, and answer "$i$ is healthier" iff $H_i > H_j$. Then

$$P(\text{answer "}i\text{ healthier"}) \;=\; \Phi\!\left(\frac{\theta_i - \theta_j}{\sqrt{2\sigma^2}}\right). \tag{1}$$

This is Thurstone's comparative-judgment model (Thurstone 1927). Intuition: states far apart in $\theta$ produce near-unanimous answers; states close together produce closer to 50/50. The observed disagreement *rates* are exactly what identifies the spacings.

**Two things the data cannot tell you** (this is what econometricians mean by an identification problem): adding a constant to every $\theta$ changes nothing, because only differences enter; multiplying all $\theta$ and $\sigma$ by the same factor changes nothing, because only the ratio enters. So we normalize by convention: set $2\sigma^2 = 1$ and fix one arbitrary reference state's $\theta$ to 0. Both choices are undone later; the test suite proves numerically that the final weights do not depend on the reference state (`test_reference_state_invariance`), and stage C exists precisely to replace the arbitrary location and scale with a meaningful one.

**Why this is "just a probit".** A probit regression models $P(y = 1) = \Phi(x'\beta)$ and finds $\beta$ by MLE. Build one row per comparison: $+1$ in the column of the first-listed state, $-1$ in the column of the second, 0 elsewhere, with the reference state's column dropped (`build_pc_design`). Then $x'\beta = \beta_{\text{first}} - \beta_{\text{second}}$, which is the model above with $\beta = \theta$. Standard probit machinery, with a globally concave log-likelihood (Newton's method reliably finds *the* maximum), estimates all the $\theta$'s at once from all comparisons pooled.

**What has to hold, and what the code does when it doesn't.** Think of states as nodes and comparisons as edges. You learn relative positions only within a connected graph; two islands of states never compared, even indirectly, have no estimable relative scale, so estimation refuses to run on a disconnected graph (`connected_components`). And a state that wins (or loses) every comparison it appears in is like an undefeated team in a league table: the data only says "better than everything it met", the likelihood keeps improving as its $\theta$ grows without bound, and there is no finite estimate. The code detects this (separation) and raises instead of returning a huge garbage number, because that number would otherwise poison stage C.

## 4 Stage B: death-anchoring questions to absolute positions

*Code: `fit_phe` in `src/welfareweights/anchor.py`.*

**The model.** A PHE answer reveals whether the respondent's valuation of state $s$ is below or above $1000/c$. Model the perceived weight on the logit scale: the respondent's draw is $L = b_s + \varepsilon$ with $\varepsilon \sim N(0, \sigma^2)$, where $b_s$ is the population logit-weight of state $s$ and $\varepsilon$ is honest disagreement between respondents. Choosing program 1 (avert deaths) reveals $L < t$ where $t = \text{logit}(1000/c)$.

Each answer is therefore a *censored* observation: you never see $L$, only which side of a known threshold it fell on. The likelihood of an answer is $\Phi((t - b_s)/\sigma)$ for one choice and its complement

for the other. With five values of $c$ the thresholds take five values, from $\text{logit}(1/10) \approx -2.20$ up to $\text{logit}(2/3) \approx 0.69$, and the observed choice frequencies across those thresholds trace out where each state's distribution sits; it is like locating a distribution by asking many people "is it below this line?" at a few different lines.

**A trick worth seeing once.** Divide through by $\sigma$ inside the $\Phi$:

$$P(\text{choose deaths program}) \;=\; \Phi\Big(t \cdot \tfrac{1}{\sigma} \;+\; \textstyle\sum_s d_s \cdot \big(-\tfrac{b_s}{\sigma}\big)\Big), \tag{2}$$

where $d_s$ is a dummy for the state asked about. The right-hand side is linear in observables (the threshold value $t$ and the state dummies), so this censored-regression problem *is* an ordinary probit of the binary choice on $[t, \text{dummies}]$ with no intercept. Fit that probit, then read off $\sigma = 1/\hat{c}_t$ and $b_s = -\hat{c}_s \cdot \sigma$. This is not an approximation; the test suite verifies it against brute-force minimization of the original likelihood (`test_probit_reduction_is_the_mle`). The payoff is the same as in stage A: a globally concave problem with dependable convergence.

**The limits of five thresholds.** A state so mild (or severe) that its entire perceived distribution sits on one side of *all* the thresholds generates unanimous answers, and unanimity is the separation problem again: the estimate diverges. This is why the instrument only asks PHE questions about mid-severity states, why the code raises on one-sided states (the pipeline instead drops them from anchoring with a warning), and why the anchor-set design matters so much (Appendix A, item 8).

## 5 Stage C: weld the two scales together

*Code: `fit_anchor_map` and `expected_expit` in `src/welfareweights/rescale.py`; chained by `estimate_dws` in `src/welfareweights/pipeline.py`.*

Stage A produced $\beta$ for every state on an arbitrary linear scale. Stage B produced $b_s = \text{logit}(\text{weight})$, an absolute death-anchored position, but only for the anchor states. If the two measurements are consistent, they must be related by a line:

$$\beta \;=\; a \cdot \text{logit}(DW) + b, \qquad a < 0 \text{ (healthier means lower weight)}. \tag{3}$$

So: over the states measured by *both* stages, run ordinary least squares of the stage-A betas on the stage-B logit-weights; invert the fitted line to convert every state's $\beta$ into a logit-weight; expit back to $(0, 1)$. That is the whole stage. The $R^2$ of that regression is one of the two most informative diagnostics in the pipeline (it measures whether the relative scale and the absolute scale actually agree), and the pipeline warns when it is low.

Three subtleties:

The line is fit on estimates, not truth, so a wildly wrong anchor point (a nearly-one-sided state that slipped through, say) can lever the whole line and bias every weight at once. This is exactly the failure our review demonstrated (max error 0.208, silently), and it is why the gates in stage B exist. An optional variant weights the regression by each anchor's precision.

The straight line itself is an assumption, and it is the load-bearing one: the misspecification battery (Appendix A, item 7) found that a bent relationship damages weight levels while leaving rankings and $R^2$ nearly intact. The pipeline therefore also fits a quadratic alongside the line

and warns when the bend is both material (more than 0.02 in weight units across the anchor range, measured as the exact line-inversion error) and statistically distinguishable from noise (`test_curvature_diagnostic` calibrates the gate). It catches severe bends reliably; mild bends sit below its power at typical anchor counts, and wider anchor sets sharpen it. The gate can only measure harm across the anchor range, so states mapped beyond it are diagnosed separately: the weights table publishes each state's extrapolation distance in anchor-span units, and the pipeline warns past half a span — the states leaning on the fitted line furthest from the data that fitted it.

The back-transform averages correctly rather than plugging in: when the logit-scale estimate has spread $\tau$ around $\mu$ (as when pooling several surveys), the reported weight is $E[\text{expit}(N(\mu, \tau^2))]$, not $\text{expit}(\mu)$; the mean of a nonlinear transform is not the transform of the mean (Jensen's inequality, largest near 0 and 1). The engine computes that expectation by Gauss–Hermite quadrature, a deterministic 64-point integral, rather than by simulation. With a single survey $\tau = 0$ and this reduces to plain expit.

## 6 Uncertainty

*Code: `bootstrap_dws` in `src/welfareweights/inference.py`.*

Each respondent answers many questions, and their answers share that respondent's quirks, so treating every row as independent overstates how much information you have and makes standard errors too small. The three-stage structure also makes textbook standard-error formulas awkward: errors from stages A and B both flow through the fitted anchor line and then through a nonlinear transform.

The engine uses the standard blunt instrument that handles all of this at once: the cluster bootstrap. Resample *respondents* (not rows) with replacement, where a drawn respondent brings all their PC and PHE answers with them; rerun the entire three-stage pipeline on each resample; read the spread of the resulting weights. The 2.5th and 97.5th percentiles across resamples form the interval. Resamples that fail estimation (a resample can lose a state or go one-sided) are counted and reported, not silently retried; many failures mean the design is too fragile at that sample size for intervals to mean much.

"Does a procedure that claims 95% actually contain the truth 95% of the time?" is an empirical question, and we test it (Appendix A, item 6).

## 7 What must be true for this to work

Assumptions, in decreasing order of how much they should worry you, with the evidence on each:

1. **The two scales are related by a straight line** (stage C's regression). This is the load-bearing wall. The misspecification battery bent the true relationship and found the pipeline's one blind spot: curvature damages weight *levels* (errors up to 0.16 at the severity tested) while leaving rankings intact and keeping $R^2$ high. The curvature gate in stage C now catches severe bends within the anchor range, and the extrapolation column flags the states that lean on the line beyond it; mild bends remain below the gate's power at typical anchor counts, so if you audit one thing, audit the anchor scatter.

2. **Respondents share a common valuation scale up to noise.** Violations we simulated (person-specific scales, within-person correlation, 5–15% of answers being coin flips, first-position bias) moved final weights barely or not at all, because they produce uniform attenuation or symmetric noise that the fitted anchor line absorbs. Rankings survived everything we threw at the estimator.

3. **The noise is normal.** Swapping in a logistic error law changed essentially nothing (rank correlation 1.000 between probit and Bradley–Terry fits). Link choice is not driving results.

4. **The death-vs-lifetime-condition trade is understood as stated.** Untestable from inside the data; it is the instrument's convention (Section 2). Real respondents may also simply refuse to trade against death; our early LLM-respondent pilots suggest exactly that failure, which is a property of respondents rather than of this estimator, but it caps what PHE-style anchoring can deliver.

## 8 How to audit this yourself

Start with the worked example: `examples/quickstart.py` simulates a small survey with known true weights, runs the full pipeline plus bootstrap, and prints truth against estimates (with intervals, coverage flags, and the anchor diagnostics) in about fifteen seconds.

Then the fast checks, in rising order of effort. Running `pytest -q` takes about a minute; what each file proves is listed in Appendix B. The studies in `studies` each rerun in minutes with recorded seeds; the reports state their designs so you can attack them (change a knob in `src/welfareweights/simulate.py`, add a violation we did not think of, try to construct data that fools the gates). And read the estimator itself: the stage modules total well under a thousand lines, mapped in `docs/PIPELINE.md`. The single most auditable property of this codebase is that the estimation logic is small.

The honest boundary of all of it: everything here validates the code against the model and probes the model's robustness. Whether real survey respondents follow anything like the model is a question only real microdata can answer (obtaining it is `ROADMAP.md` item 1), and the held-out checks in `holdout_fit` are built so the same audit transfers to that data the day it arrives.

## A Validation evidence (the referee-facing summary)

Every number is reproducible from committed code with recorded seeds; file references give the exact provenance.

1. **The code recovers known truth** (`tests/test_recovery.py`, `tests/test_survey.py`). Simulating from the exact assumed DGP with known weights, the full pipeline recovers them: correlation $> 0.995$, max error $< 0.04$ at the tested sample sizes, through both the raw-table path and the instrument-assignment/answer-parsing round trip. The stage-B probit reduction is verified to be the exact MLE against direct minimization; the back-transform integral is verified against Monte Carlo.

2. **Arbitrary choices don't move the estimates** (`tests/test_metamorphic.py`). Reference state, state labels, presentation side (with response flipped), the common scale of

deaths/cases, and dataset duplication all leave weights unchanged at tolerances between $10^{-5}$ and $10^{-9}$.

3. **Failure modes are loud** (`tests/test_failure_modes.py`). Separation in either stage, non-convergence, one-sided anchor states, a degenerate or reversed anchor map, a disconnected comparison graph, contradictory deaths figures, and self-paired rows all raise or warn. The silent alternative is real: an unguarded run with weakly identified anchors produced max error 0.208 while showing rank correlation 0.99.

4. **Results are not a link-function artifact** (`tests/test_crosscheck.py`). Bradley–Terry (logit) on the identical design: Spearman 1.000, standardized-coefficient correlation 0.99998.

5. **The response model fits out of sample** (`src/welfareweights/validate.py`). Holding out 30% of respondents: held-out log-likelihood per response $-0.167$ vs. $-0.693$ null (paired comparisons) and $-0.321$ vs. $-0.690$ (anchoring), calibration slopes 0.996 and 0.993 against the ideal 1.

6. **Intervals have their stated coverage** (`studies/RESULTS-coverage.md`). In a 60-replication Monte Carlo, 95% percentile intervals cover at 0.917 (normal intervals 0.935), inside the pre-stated $[0.88, 0.99]$ band; zero failed replicates in 6,000 bootstrap fits.

7. **Rankings survive every misspecification tested; levels survive all but one** (`studies/RESULTS-misspecification.md`). Under scale heterogeneity, within-respondent correlation, 5–15% lapses, position bias, or a logistic error law, Spearman stays at 1.000 and max level error stays within 0.012 of the 0.019 baseline. The exception is curvature in the utility link: level errors up to 0.161 with rank correlation 0.9998 and $R^2$ 0.985. The pipeline now gates this (quadratic fit alongside the line; warn when material and significant; calibration in `tests/test_curvature_diagnostic.py`: fired in every severe-curvature rep at $p \leq 0.005$, silent on clean data with $p$ from 0.12 to 0.86 across sample sizes). Mild bends sit below the gate's power at typical anchor counts.

8. **Precision scales as theory predicts** (`studies/RESULTS-power.md`). Log-log slope of mean error on respondent count: $-0.517$ ($r^2$ 0.984), the root-$N$ rate. Widening the anchor set cuts mean error 44% at zero added responses; 500 respondents at 10 pairs meet a 0.05 max-error bar, 2,000 meet 0.02; the cheap precision proxy agrees with the full bootstrap to 1%.

9. **The published survey scale is comfortably sufficient** (`studies/RESULTS-gbd-scale.md`). At the GBD 2010 design size (220 states, 14,000 respondents, 15 pairs each, 30 anchors): rank correlation 0.9997, mean error 0.0047, max error 0.039; extrapolation error concentrates in the upper-middle deciles, as the logit geometry predicts.

**What this does not show.** Synthetic validation proves the code against the model and maps the model's robustness; it cannot prove the model against real respondents. That requires the original microdata (being requested) or new survey data. Until then the claims are deliberately limited: the implementation is correct, and the estimator's failure conditions are characterized and gated.

## B Reproduction

From a checkout with the virtual environment installed (`pip install -e .[dev]`):

- `python examples/quickstart.py`: the end-to-end demonstration (∼15 s).
- `pytest -q`: the full suite (∼1 min). By file: `tests/test_recovery.py` (known-truth recovery; stage-B reduction is the MLE), `tests/test_metamorphic.py` (invariances), `tests/test_failure_modes.py` (every hazard raises or warns), `tests/test_curvature_diagnostic.py` (the $R^2$-evading hazard is gated), `tests/test_crosscheck.py` (link robustness), `tests/test_parsers.py` (free-text answer contract), `tests/test_validate.py` (held-out fit machinery), `tests/test_inference.py` (bootstrap smoke test, label-immunity and support-floor regressions), `tests/test_input_checks.py` (malformed input fails loudly, naming frame, column, and cause), `tests/test_extrapolation.py` (weights beyond the anchor range are measured and warned about).
- `python studies/coverage.py`, `misspecification.py`, `power_curves.py`, `gbd_scale.py`: the four studies (minutes each; seeds recorded in each script and report).

## References


[Salomon et al.(2012)] Salomon, J.A., et al. (2012). "Common values in assessing health outcomes from disease and injury: disability weights measurement study for the Global Burden of Disease Study 2010." *The Lancet* 380(9859):2129–2143. doi:10.1016/S0140-6736(12)61680-8. Estimation methodology in the supplementary appendix; aggregate weights via GHDx.

[Salomon et al.(2015)] Salomon, J.A., et al. (2015). "Disability weights for the Global Burden of Disease 2013 study." *The Lancet Global Health* 3(11):e712–e723. doi:10.1016/S2214-109X(15)00069-8.

[Robinson et al.(2019)] Robinson, L.A., Hammitt, J.K., et al. (2019). *Reference Case Guidelines for Benefit-Cost Analysis in Global Health and Development.* Harvard T.H. Chan School of Public Health. media.repository.chds.hsph.harvard.edu (stable repository copy).

[HHS(2016)] U.S. Department of Health and Human Services, Office of the Assistant Secretary for Planning and Evaluation (2016). *Guidelines for Regulatory Impact Analysis.* aspe.hhs.gov/reports/guidelines-regulatory-impact-analysis.

[Thurstone(1927)] Thurstone, L.L. (1927). "A law of comparative judgment." *Psychological Review* 34(4):273–286. doi:10.1037/h0070288.