Package: nutrigon (via r-universe)

November 7, 2026

Title Angular Separation Between Nutritional Optima

Version 0.1.0

Description Fits response surfaces (ordinary least squares, generalized linear models with any family including quasi-families, or linear mixed-effects models) to trait data collected under the Geometric Framework for Nutrition and computes the angular separation (theta) between the nutritional optima of two traits, with percentile bootstrap confidence intervals that propagate surface-estimation uncertainty (case or stratified resampling; paired or unpaired). Provides a design-matched coverage check that simulates validation configurations on the user's own design and at the signal-to-noise ratio of the fitted surfaces, an equivalence test for alignment of optima, local Gaussian and mean curvature, and a surface-area ratio. Includes the Drosophila lifespan and fecundity dataset of Lee et al. (2008) ``Lifespan and reproduction in Drosophila: new insights from nutritional geometry'' <doi:10.1073/pnas.0710787105> as a worked example. The angular approach was introduced in Morimoto (2023) ``Nutrigonometry I: using right-angle triangles to quantify nutritional trade-offs in performance landscapes'' <doi:10.1086/723599>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.1)

Imports stats, grDevices, withr

Suggests ggplot2, ggdendro, ggrepel, patchwork, knitr, rmarkdown, testthat (>= 3.0.0), lme4, lmerTest, pracma

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/jmor2753/nutrigon

BugReports https://github.com/jmor2753/nutrigon/issues

NeedsCompilation no

Author Juliano Morimoto [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3561-1920>)

Maintainer Juliano Morimoto <juliano.morimoto@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-07 16:57:13 UTC

RemoteUrl https://github.com/cran/nutrigon

RemoteRef HEAD

RemoteSha 36eb0b9df9c5ebb5ea4337021b7809299e60d36b

Contents

lee_drosophila 2
nutrigon_build_grid 3
nutrigon_check 6
nutrigon_coefs 8
nutrigon_compare 9
nutrigon_curvature 12
nutrigon_elongation 13
nutrigon_equivalence 14
nutrigon_fit 15
nutrigon_grid 18
nutrigon_hull 19
nutrigon_matrix 20
nutrigon_region_polygon 22
nutrigon_snr 23
nutrigon_validation_grid 24
zanco_survival 25
zanco_tradeoff_matrix 26
zanco_wingsize 27

Index **28**

`lee_drosophila` *Lee et al. (2008) Drosophila lifespan and fecundity data*

Description

Nutrient intake, lifespan, and daily egg production for female *Drosophila melanogaster* reared across a geometric array of protein:carbohydrate diets. Used throughout the package examples and vignette as a worked, paired comparison (both traits measured on the same individual flies).

Usage

```
lee_drosophila
```

Format

A data frame with 973 rows (complete cases) and 11 columns:

carb_eaten Carbohydrate eaten (mg).

protein_eaten Protein eaten (mg).

lifespan Lifespan (days).

lifetimeegg Lifetime egg count.

dailyeggs Daily egg production rate.

treatment Design-point identifier (diet treatment code); used for within-design-point bootstrap resampling.

replicate Replicate identifier within treatment.

Ratio Nominal protein:carbohydrate ratio, as a string, e.g. `"(0:1)"`.

Food Total food concentration.

P_fixed Fixed protein concentration of the diet.

C_fixed Fixed carbohydrate concentration of the diet.

Source

Lee, K.P., Simpson, S.J., Clissold, F.J., Brooks, R., Ballard, J.W.O., Taylor, P.W., Soran, N. & Raubenheimer, D. (2008). Lifespan and reproduction in Drosophila: new insights from nutritional geometry. *PNAS*, 105(7), 2498-2503. doi:10.1073/pnas.0710787105

Included with the original authors' permission. Please cite the original paper when using these data.

`nutrigon_build_grid` *Build a validation grid for the design-matched coverage check*

Description

Simulates many two-surface configurations with known true optima, runs the full bootstrap on repeated simulated datasets for each, and records the empirical coverage of the interval. The grid is used by `nutrigon_check()` to estimate the coverage expected for a specific result.

Usage

```
nutrigon_build_grid(
  n_grid_points = 100,
  n_sim_per_point = 20,
  B_boot = 50,
  design = NULL,
  x_col = NULL,
  y_col = NULL,
  design_id_col = NULL,
  snr = NULL,
  paired = TRUE,
  radius_quantiles = c(0.25, 0.95),
  elongation_sampler = function() stats::runif(1, 1, 40),
  rail_ratios = c(0.1, 0.3, 1, 3, 10),
  n_levels = 3,
  intake_sd = 8,
  response_sd = 2,
  n_cloud = 100,
  n_reps_cloud = 3,
  seed = 1,
  verbose = TRUE,
  n = NULL,
  n_sim = NULL,
  theta_min = NULL,
  theta_max = NULL
)
```

Arguments

`n_grid_points`	Number of configurations (default 100).
`n_sim_per_point`	Simulated datasets per configuration (default 20; coverage per configuration is estimated in steps of `1 / n_sim_per_point`).
`B_boot`	Bootstrap resamples per simulated dataset (default 50).
`design`	Optional data frame of your own data (switches on the design-matched mode).
`x_col`, `y_col`, `design_id_col`	Column names in `design` (required with `design`).
`snr`	Optional target signal-to-noise ratio(s), one or two positive numbers (surface 1, surface 2), e.g. from `nutrigon_snr()`.
`paired`	`TRUE` (default) to simulate both traits on the same individuals; `FALSE` to simulate them on independent individuals and resample them independently.
`radius_quantiles`	Quantiles of the observed radial intake between which true optima are placed (design-matched mode).
`elongation_sampler`	Function of no arguments returning one elongation ratio (default: uniform on 1-40).

`rail_ratios`, `n_levels`, `intake_sd`, `response_sd`
: Settings of the generic mode. `response_sd` is ignored when `snr` is supplied.

`n_cloud`, `n_reps_cloud`
: Monte Carlo fallback settings passed to `nutrigon_compare()`.

`seed`
: Random seed (set locally).

`verbose`
: Print progress.

`n`, `n_sim`, `theta_min`, `theta_max`
: Deprecated (version 0.1.0 names). `n` and `n_sim` map onto `n_grid_points` and `n_sim_per_point`; `theta_min`/`theta_max` are ignored (the support is computed from the design).

Value

A data frame with one row per configuration: `max_elongation`, `min_edge_distance`, `angle_diff`, `coverage`, `n_valid_sims`, `theta1`, `theta2`, `radius1`, `radius2`. Attributes: `angular_support`, `grid_design` (`"user_design"` or `"default_rails"`), `snr`, `paired`.

Two modes

Design-matched (recommended; `design = your data`**)** Configurations are simulated on your own experimental design: the observed design points (mean intake of each `design_id_col` group) with the observed number of replicates per point; the observed within-point standard deviations of intake as intake noise; true optima at angles drawn uniformly within the angular support of your design hull and at radial distances drawn uniformly between the `radius_quantiles` of the observed radial intakes (accepted only if inside the hull). If `snr` is supplied (see `nutrigon_snr()`), response noise is set so that each simulated surface has that signal-to-noise ratio.

Generic (`design = NULL`) A generic rail design (`rail_ratios` x `n_levels`, one observation per design point, optima at radial distances 145 and 90, intake noise `intake_sd`, response noise `response_sd`), as in the original validation.

All configurations are drawn before any simulation, so grids built with the same `seed` but different `snr` or `paired` settings contain identical configurations and can be compared configuration by configuration. Elongation ratios are drawn by `elongation_sampler`.

This function is slow: it runs `n_grid_points * n_sim_per_point` bootstraps of `B_boot` replicates each. Build once, save with `saveRDS()`, and reuse.

See Also

`nutrigon_check()`, `nutrigon_snr()`

Examples

```
data(lee_drosophila)
# Tiny grid, for illustration only (use the defaults in practice)
fit1 <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten", "lifespan")
fit2 <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten", "dailyeggs")
grid <- nutrigon_build_grid(
  n_grid_points = 3, n_sim_per_point = 3, B_boot = 20,
```

```
  design = lee_drosophila, x_col = "protein_eaten", y_col = "carb_eaten",
  design_id_col = "treatment",
  snr = c(nutrigon_snr(fit1), nutrigon_snr(fit2)), seed = 99)
grid
```

`nutrigon_check` *Design-matched coverage check for one result*

Description

Estimates the coverage expected for the interval of a specific result by averaging the simulated coverage of the k validation-grid configurations closest to it in three risk factors: maximum curvature elongation, minimum distance of either optimum from the edge of its design's angular support, and the angle difference. Distances are standardised by the grid's own standard deviations.

Usage

```
nutrigon_check(
  result,
  elong1,
  elong2,
  k = 10,
  validation_grid = "default",
  min_angle = 5,
  conf_level = 0.95,
  theta_min = 5.71,
  theta_max = 84.29
)
```

Arguments

`result`	A `"nutrigon_comparison"` object from `nutrigon_compare()`.
`elong1, elong2`	Elongation ratios of the two surfaces (from `nutrigon_elongation()`).
`k`	Number of nearest grid configurations (default 10).
`validation_grid`	A grid from `nutrigon_build_grid()`; `"default"` (the shipped generic grid); or `NULL` to build a small generic grid on the fly (slow; for testing only).
`min_angle`	Near-zero threshold in degrees (default 5).
`conf_level`	Confidence level of the Wilson interval for the estimated coverage (default 0.95).
`theta_min, theta_max`	Angular support to use for the edge distance only if `result` does not carry its own (objects from version 0.1.0).

Value

An object of class "nutrigon_check", a list with:

flag Plain-language summary.

estimated_coverage Mean simulated coverage of the k nearest eligible configurations (NA for a near-zero result).

coverage_ci Wilson interval for estimated_coverage, pooling the simulated datasets of the neighbours (NA if the grid has no n_valid_sims column).

n_sims_pooled Number of simulated datasets behind coverage_ci.

risk_factors The result's own risk factors.

n_neighbours Number of neighbours used.

n_grid_points_eligible Grid configurations with an angle difference of at least min_angle.

neighbours, neighbour_distances The neighbouring grid rows and their standardised distances.

Recommended use

Build the grid on your own design and at the signal-to-noise ratio of your fitted surfaces with nutrigon_build_grid() (design = your data, snr = c(nutrigon_snr(fit1), nutrigon_snr(fit2))). The grid shipped with the package (validation_grid = "default") was simulated on a generic 5-rail design with fixed radial positions and noise; a warning is issued when its angular support differs from that of your result.

Near-zero angles

Near zero, the unsigned percentile interval cannot contain the true value by construction. Grid configurations with an angle difference below min_angle are therefore excluded from the neighbour search, and results with theta_hat < min_angle are not given a coverage estimate. To assess whether two optima are aligned, use nutrigon_equivalence().

Limitations

The check is conditional on the grid's generating model (quadratic surfaces with Gaussian noise), does not account for model misspecification, and is only as fine-grained as the grid near the result. It is a diagnostic and does not change the interval.

See Also

nutrigon_build_grid(), nutrigon_compare(), nutrigon_equivalence()

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
fit_eggs <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                         "dailyeggs", design_id_col = "treatment")
result <- nutrigon_compare(
  lee_drosophila, "protein_eaten", "carb_eaten", "lifespan",
```

```
  lee_drosophila, "protein_eaten", "carb_eaten", "dailyeggs",
  paired = TRUE, B = 50, seed = 1
)
# Quick look with the shipped generic grid (warns: different design)
check <- suppressWarnings(
  nutrigon_check(result, elong1 = nutrigon_elongation(fit_lifespan),
                 elong2 = nutrigon_elongation(fit_eggs)))
check


# Recommended: a grid matched to the design and signal-to-noise ratio
grid <- nutrigon_build_grid(
  n_grid_points = 10, n_sim_per_point = 5, B_boot = 20,   # use defaults in practice
  design = lee_drosophila, x_col = "protein_eaten", y_col = "carb_eaten",
  design_id_col = "treatment",
  snr = c(nutrigon_snr(fit_lifespan), nutrigon_snr(fit_eggs)), seed = 99)
nutrigon_check(result, nutrigon_elongation(fit_lifespan),
               nutrigon_elongation(fit_eggs), validation_grid = grid)
```

`nutrigon_coefs` *Extract fitted quadratic coefficients from a* `nutrigon_fit` *object*

Description

Convenience accessor for the fitted link-scale quadratic coefficients (`.x`, `.y`, `I(.x^2)`, `I(.y^2)`, `.x:.y`), regardless of whether the underlying model is `"lm"`, `"glm"`, or `"lmer"`. Use this instead of calling `coef()` on `fit$model` directly, since for `"lmer"` fits `fit$model` is a wrapper object, not the raw `merMod`, and for `"glm"` fits the coefficients are on the link scale by design (see `nutrigon_fit()`).

Usage

```
nutrigon_coefs(fit)
```

Arguments

`fit`	An object of class `"nutrigon_fit"`.

Value

A named numeric vector.

See Also

`nutrigon_fit()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
nutrigon_coefs(fit_lifespan)
```

`nutrigon_compare` *Compare two nutritional optima and bootstrap the angular separation*

Description

Fits two quadratic response surfaces, locates each nutritional optimum with the hybrid estimator (closed-form stationary point when admissible, otherwise a constrained Monte Carlo search with the `max` rule), and computes the angular separation theta between the two optima together with a percentile bootstrap confidence interval. Each bootstrap replicate refits both surfaces, so uncertainty in the surface coefficients is propagated into the location of each optimum and into theta.

Usage

```
nutrigon_compare(
  data1,
  x1_col,
  y1_col,
  z1_col,
  data2,
  x2_col,
  y2_col,
  z2_col,
  design_id_col1 = NULL,
  design_id_col2 = NULL,
  paired = TRUE,
  resample = c("case", "stratified"),
  peak1 = TRUE,
  peak2 = TRUE,
  model_type1 = c("lm", "glm", "lmer"),
  model_type2 = c("lm", "glm", "lmer"),
  re_formula1 = NULL,
  re_formula2 = NULL,
  re_group_cols1 = NULL,
  re_group_cols2 = NULL,
  family1 = NULL,
  family2 = NULL,
  extra_args1 = list(),
  extra_args2 = list(),
  B = 500,
  conf_level = 0.95,
```

```
  n_cloud = 500,
  n_reps_cloud = 5,
  seed = NULL
)
```

Arguments

data1, data2	Data frames for traits 1 and 2. Can be the same object if both traits were measured on the same individuals.
x1_col, y1_col, z1_col	Column names for surface 1.
x2_col, y2_col, z2_col	Column names for surface 2.
design_id_col1, design_id_col2	Design-point ID columns. Required for resample = "stratified"; otherwise optional.
paired	TRUE (default) if both traits were measured on the same individuals (rows of data1 and data2 must correspond); FALSE for unpaired data.
resample	"case" (default, validated) or "stratified". See the Resampling section.
peak1, peak2	TRUE to locate maxima; FALSE for minima.
model_type1, model_type2	One of "lm" (default), "glm", or "lmer", for surfaces 1 and 2 respectively. See nutrigon_fit().
re_formula1, re_formula2	Random-effects formula strings, required only when the corresponding model_type is "lmer".
re_group_cols1, re_group_cols2	Grouping column name(s) referred to by the corresponding re_formula.
family1, family2	Forwarded as family = to stats::glm() when the corresponding model_type is "glm".
extra_args1, extra_args2	Named lists of further arguments for the underlying fitters (e.g. list(weights = ...)).
B	Number of bootstrap resamples (default 500).
conf_level	Confidence level of the percentile interval (default 0.95).
n_cloud	Monte Carlo fallback cloud size.
n_reps_cloud	Number of Monte Carlo draws per estimate.
seed	Optional random seed (set locally; the global RNG state is restored on exit).

Value

An object of class "nutrigon_comparison", a list with:

theta_hat Point estimate of angular separation (degrees), |theta1 - theta2|.

ci Percentile bootstrap CI for theta (unsigned), named e.g. c(lower_95, upper_95).

theta1, theta2 Polar angles of the two optima (degrees).

point1, point2 Locations of the two optima, `c(x, y)`, in the original nutrient units.

used_fallback1, used_fallback2 Whether the Monte Carlo fallback located each full-data optimum.

angular_support List with `surface1` and `surface2`, each `c(theta_min, theta_max)`: the angular range of that surface's design hull (degrees). Used by `nutrigon_check()`.

boot_diffs The `B` bootstrap values of theta (`NA` for failed replicates).

boot_signed The `B` bootstrap values of `theta1 - theta2`.

fallback_rate Proportion of successful bootstrap replicates in which at least one optimum required the Monte Carlo fallback.

n_failed Number of bootstrap replicates that failed (e.g. a rank-deficient refit) and were excluded.

n_fallback Deprecated alias of `n_failed`, kept for compatibility with version 0.1.0.

B, conf_level, paired, resample Settings used.

Resampling

`resample = "case"` (default) draws observations (rows) with replacement from the whole dataset. This is the scheme whose coverage was validated by simulation. With `paired = TRUE` the same rows are used for both traits, which preserves the covariance between the two fitted surfaces when both traits are measured on the same individuals; with `paired = FALSE` each trait is resampled independently. `resample = "stratified"` resamples within each design point (requires `design_id_col1`/`design_id_col2`), keeping the number of observations per design point fixed. It is useful for balanced designs with several replicates per design point, but its coverage has not been separately validated.

What theta measures

theta is the difference between the polar angles of the two optima, i.e. the difference in nutrient balance (e.g. protein:carbohydrate direction). It does not measure the radial distance of the optima from the origin (total intake) or the fitness cost of feeding between them. Report it with the locations of the optima (`point1`, `point2`).

See Also

`nutrigon_fit()`, `nutrigon_check()`, `nutrigon_equivalence()`

Examples

```
data(lee_drosophila)
result <- nutrigon_compare(
  lee_drosophila, "protein_eaten", "carb_eaten", "lifespan",
  lee_drosophila, "protein_eaten", "carb_eaten", "dailyeggs",
  design_id_col1 = "treatment", design_id_col2 = "treatment",
  paired = TRUE, B = 50, seed = 1
)
print(result)
```

```
# Different model types for the two surfaces
result_glm <- nutrigon_compare(
  lee_drosophila, "protein_eaten", "carb_eaten", "lifespan",
  lee_drosophila, "protein_eaten", "carb_eaten", "dailyeggs",
  paired = TRUE, B = 50, seed = 1,
  model_type2 = "glm", family2 = quasipoisson()
)
```

`nutrigon_curvature` *Local Gaussian and mean curvature, and surface-area ratio, of a fitted surface*

Description

Computes the local Gaussian curvature and mean curvature of a fitted quadratic surface at every point of a grid restricted to the convex hull of the design, together with a surface-area ratio (the fitted surface's area relative to a flat surface spanning the same design). This treats the fitted response surface as a 2-manifold embedded in 3-space (the first and second fundamental forms of differential geometry), which is a different notion of "curvature" from the Hessian-eigenvalue elongation ratio returned by `nutrigon_elongation()` – elongation describes the *shape* of the quadratic (how stretched the peak/valley is along its principal axes), while Gaussian/mean curvature describe the *local bending* of the surface itself at each point.

Usage

```
nutrigon_curvature(fit, n = 50, quiet = FALSE)
```

Arguments

`fit`	A `"nutrigon_fit"` object (from `nutrigon_fit()`).
`n`	Grid resolution per axis (default 50, matching the resolution used in the original curvature analysis this function is based on). Only points inside the convex hull of the design are returned.
`quiet`	Suppress the link-scale reminder message for GLM fits (default `FALSE`).

Details

For `model_type = "glm"` fits (see `nutrigon_fit()`), curvature is computed on the **link scale**, consistent with how the optimum itself is located and how `nutrigon_elongation()` computes elongation. The fitted quadratic's coefficients are read via the same internal dispatch layer used throughout the package, so curvature is available identically for `"lm"`, `"glm"`, and `"lmer"` fits.

Gaussian curvature (`Gauss_curv`) is the product of the two principal curvatures at a point; mean curvature (`Mean_curv`) is their average. Both are computed in closed form from the first and second fundamental forms of the fitted quadratic surface, following standard differential geometry (see e.g. do Carmo, *Differential Geometry of Curves and Surfaces*).

The surface-area ratio (`surface_area_ratio`, a single number returned alongside the grid, not per-point) is the fitted surface's area over the design hull, divided by the area of a flat surface spanning the same hull. A ratio of 1 indicates a flat response; larger values indicate a more strongly curved (bowl-, dome-, or saddle-shaped) surface. This is computed by numerically integrating the surface-area element (via **pracma**) over the bounding box of the design, for both the fitted surface and a flat reference surface fitted to a constant response over the same design.

Value

A list with components:

grid A data frame with columns named after the original `x_col` and `y_col`, plus `Gauss_curv` and `Mean_curv`, one row per grid point inside the design hull.

surface_area_ratio A single number: the fitted surface's area divided by a flat reference surface's area over the same design bounding box (see Details). Requires **pracma**.

See Also

`nutrigon_fit()`, `nutrigon_elongation()`, `nutrigon_grid()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
curv_lifespan <- nutrigon_curvature(fit_lifespan, n = 30)
head(curv_lifespan$grid)
curv_lifespan$surface_area_ratio
```

---

nutrigon_elongation *Curvature elongation ratio of a fitted quadratic surface*

Description

Returns the ratio of the larger to smaller absolute eigenvalue of the fitted quadratic's Hessian (on the link scale for GLM fits – see Details). A value near 1 indicates roughly symmetric curvature; large values (e.g. > 10) indicate a narrow, ridge-like peak. This is used as one of the risk factors in `nutrigon_check()`.

Usage

```
nutrigon_elongation(fit)
```

Arguments

`fit`	A `"nutrigon_fit"` object (from `nutrigon_fit()`); or a `nutrigon_*_fit` wrapper object (i.e. `some_fit$model`) for `"lm"`, `"glm"`, or `"lmer"` fits.

Details

For `model_type = "glm"` fits, the Hessian is computed on the **link scale**, consistent with how the optimum itself is located (see `?nutrigon_fit`). This means elongation values are directly comparable between `"lm"` and `"glm"` fits only insofar as both describe curvature on the scale each model actually estimates a quadratic on – they are not automatically on a common footing with an `"lm"` fit of the same trait's raw values.

Value

A single positive number (the elongation ratio, >= 1), or `NA` if the Hessian is singular.

See Also

`nutrigon_fit()`, `nutrigon_check()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
nutrigon_elongation(fit_lifespan)
```

`nutrigon_equivalence` *Equivalence test for alignment of two nutritional optima*

Description

Because the angular separation is non-negative, its percentile interval lies above zero even when the true separation is zero, so it cannot by itself show that two optima are aligned. This function implements the equivalence approach: given a smallest angular separation of biological interest `delta`, chosen before looking at the data, it tests $H_0 : \theta \geq \delta$ against $H_1 : \theta < \delta$ and rejects H_0 (concluding equivalence, i.e. alignment within `delta`) when the upper `1 - alpha` percentile of the bootstrap distribution of theta lies below `delta`.

Usage

```
nutrigon_equivalence(result, delta, alpha = 0.05)
```

Arguments

`result`	A `"nutrigon_comparison"` object from `nutrigon_compare()`.
`delta`	Smallest angular separation of biological interest (degrees), specified in advance.
`alpha`	Significance level (default 0.05; the bound is the upper limit of a `1 - 2 * alpha` two-sided percentile interval).

Details

The estimator is biased upwards near zero, which inflates the upper bound, so the procedure errs towards not declaring alignment. Choose `delta` above the near-zero zone in which coverage is poor (about 5 degrees at realistic signal-to-noise ratios). The calibration of this one-sided bound has not been validated separately by simulation.

Value

A list with `upper_bound`, `delta`, `alpha`, `equivalent` (logical) and a plain-language `conclusion`.

References

Schuirmann, D.J. (1987) A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics* 15: 657-680.

See Also

`nutrigon_compare()`, `nutrigon_check()`

Examples

```
data(lee_drosophila)
result <- nutrigon_compare(
  lee_drosophila, "protein_eaten", "carb_eaten", "lifespan",
  lee_drosophila, "protein_eaten", "carb_eaten", "dailyeggs",
  paired = TRUE, B = 50, seed = 1)
nutrigon_equivalence(result, delta = 10)
```

---

nutrigon_fit *Fit a quadratic response surface and locate the nutritional optimum*

Description

Fits a full quadratic response surface (`z ~ x + y + x^2 + y^2 + x:y`) to trait data collected under the Geometric Framework for Nutrition, and locates the nutritional optimum (peak or valley) of the fitted surface. The optimum is found in closed form when the fitted surface has the correct curvature and the stationary point falls inside the convex hull of the design; otherwise a Monte Carlo fallback samples points inside the hull and takes the top (or bottom) quantile region.

Usage

```
nutrigon_fit(
  data,
  x_col,
  y_col,
  z_col,
```

```
  design_id_col = NULL,
  model_type = c("lm", "glm", "lmer"),
  re_formula = NULL,
  re_group_cols = NULL,
  peak = TRUE,
  n_cloud = 500,
  n_reps = 5,
  region_quantile = NULL,
  ...
)
```

Arguments

`data`	A data frame containing nutrient intake and trait columns.
`x_col`	Name of the protein (x-axis nutrient) column.
`y_col`	Name of the carbohydrate (y-axis nutrient) column.
`z_col`	Name of the trait (response) column.
`design_id_col`	Name of a column identifying distinct design points (e.g. treatment, or rail x concentration combination). Stored with the fit for reference; design-point identifiers are used by `nutrigon_compare()` only with `resample = "stratified"` and by `nutrigon_build_grid()` for design-matched grids.
`model_type`	One of `"lm"` (default), `"glm"`, or `"lmer"`. See Details.
`re_formula`	For `model_type = "lmer"` only: the random-effects portion of the model formula, as a string starting with `"+"`, e.g. `"+ (1 | block)"`. Required when `model_type = "lmer"`; ignored otherwise. Any valid **lme4** random-effects syntax is supported – multiple/crossed terms (`"+ (1 | block) + (1 | batch)"`), nesting (`"+ (1 | block/subblock)"`), interaction grouping factors (`"+ (1 | block:batch)"`), and random slopes – since this string is passed to **lme4** essentially unmodified. **For random slopes**, refer to the nutrient columns as `.x` and `.y` (not your original `x_col`/`y_col` names), e.g. `"+ (1 + .x | block)"`: internally the fitting data frame renames your columns to `.x`/`.y`/`.z` before fitting, so `"+ (1 + x | block)"` will fail with an `object 'x' not found` error if your column was actually named `x_col = "protein"` or similar. Random *intercepts* (`"+ (1 | block)"`) are unaffected, since they don't reference the nutrient columns by name.
`re_group_cols`	For `model_type = "lmer"` only: character vector of column name(s) in `data` that `re_formula` refers to as grouping factors (e.g. `"block"` for `re_formula = "+ (1 | block)"`; both `c("block", "batch")` for crossed terms or `c("block", "subblock")` for nested terms). Required whenever `re_formula` is supplied. Needed because internally the data is reduced to just the x/y/z columns before fitting; any column `re_formula` needs must be named here so it is carried through.
`peak`	`TRUE` (default) to locate the maximum; `FALSE` for the minimum.
`n_cloud`	Number of candidate points for the Monte Carlo fallback.
`n_reps`	Number of MC draws (the final region is the union of all draws).

`region_quantile`
: Top (or bottom) quantile defining the peak region. Default: 0.95 for peaks, 0.05 for valleys.

`...`
: Forwarded to the underlying fitter: e.g. `family = binomial()` or `family = quasipoisson()` for `model_type = "glm"`; `weights = , control = ,` etc. for `model_type = "glm"` or `"lmer"`. Ignored for `model_type = "lm"`.

Details

Three model families are supported via `model_type`:

`"lm"` (default) Ordinary least squares. Appropriate for approximately Gaussian, homoscedastic trait responses.

`"glm"` Generalized linear model, any `family` (including quasi-families such as `quasipoisson()`, `quasibinomial()` for overdispersed counts/proportions). Pass `family =` via `....` The closed-form and Monte Carlo optimum-finding both operate on the **link scale** (see Details), which is the scale on which the surface is quadratic.

`"lmer"` Linear mixed-effects model via **lme4**. Requires `re_formula` specifying the random-effects structure, e.g. `re_formula = "+ (1 | block)"`. The optimum located is the population-level optimum (random effects integrated out via `re.form = NA`), not a group-specific optimum.

For `model_type = "glm"`, the fitted linear predictor $\eta(x, y) = \beta_0 + \beta_x x + \beta_y y + \beta_{xx} x^2 + \beta_{yy} y^2 + \beta_{xy} xy$ is quadratic in (x, y), but the predicted trait value on the response scale generally is not (e.g. for a logit or log link). Because the link function is monotonic, the point that maximises η also maximises the response-scale prediction, so locating the optimum on the link scale and reporting it in the original nutrient units loses nothing, and lets the same closed-form stationary-point algebra be reused unchanged across `"lm"`, `"glm"`, and `"lmer"`.

Value

An object of class `"nutrigon_fit"`, a list with components:

optimum Data frame with `x`, `y`, `theta` (angle from origin, degrees), and `used_fallback` (logical).

model A `nutrigon_*_fit` wrapper object. The raw fitted `lm`/`glm`/`merMod` object is available as `model$fit` (e.g. for `summary()`, residual diagnostics, or `AIC()`).

model_type Character: `"lm"`, `"glm"`, or `"lmer"`.

data The cleaned input data.

hull Design hull vertices (for plotting).

region The top/bottom quantile region (for plotting), on the link scale for `model_type = "glm"`.

elongation Curvature elongation ratio (eigenvalue ratio of the Hessian), computed on the link scale.

See Also

`nutrigon_compare()`, `nutrigon_elongation()`, `nutrigon_coefs()`

Examples

```
data(lee_drosophila)

# Default: ordinary least squares
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
print(fit_lifespan)


# GLM example: a count-like trait fitted with a quasi-Poisson family
fit_eggs_glm <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "dailyeggs", design_id_col = "treatment",
                             model_type = "glm", family = quasipoisson())

# Mixed-model example (requires lme4 and lmerTest; install with
# install.packages(c("lme4", "lmerTest")) if not already available).
# re_group_cols must name the original column(s) re_formula refers to,
# since the fitter only sees the x/y/z columns by default.
fit_lifespan_lmer <- nutrigon_fit(
  lee_drosophila, "protein_eaten", "carb_eaten", "lifespan",
  model_type = "lmer", re_formula = "+ (1 | treatment)",
  re_group_cols = "treatment"
)
```

`nutrigon_grid` *Build a prediction grid inside the design hull for surface plotting*

Description

Constructs an evenly-spaced grid over the range of the fitted data, restricted to points inside the convex hull of the design, with model predictions attached. Intended for use with `ggplot2::geom_raster()`.

Usage

```
nutrigon_grid(fit, n = 120, quiet = FALSE)
```

Arguments

`fit`	A `"nutrigon_fit"` object (from `nutrigon_fit()`).
`n`	Grid resolution per axis (default 120).
`quiet`	Suppress the link-scale reminder message for GLM fits (default `FALSE`).

Value

A data frame with columns named after the original `x_col` and `y_col`, plus `predicted`. Only points inside the convex hull are included.

GLM fits

when `fit` was produced with `model_type = "glm"` (see `nutrigon_fit()`), the `predicted` column is on the **link scale** (e.g. logit(p) for a binomial GLM, log(mu) for a Poisson/ quasi-Poisson GLM), not the original response scale. This is deliberate and consistent with how the optimum itself was located (see `?nutrigon_fit`, Details). A message is printed reminding you of this every time `nutrigon_grid()` is called on a GLM fit; pass `quiet = TRUE` to suppress it. If you want a response-scale surface for plotting (e.g. predicted probabilities rather than logit-odds), back-transform the `predicted` column yourself with the appropriate inverse link, e.g. `plogis()` for a logit link or `exp()` for a log link – see Examples.

See Also

`nutrigon_hull()`, `nutrigon_region_polygon()`, `nutrigon_fit()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
grid_lifespan <- nutrigon_grid(fit_lifespan, n = 40)
head(grid_lifespan)


# GLM example: predicted is on the logit scale; back-transform for plots
# of predicted probability rather than logit-odds
fit_glm <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                        "dailyeggs", model_type = "glm",
                        family = quasipoisson())
grid_glm <- nutrigon_grid(fit_glm, n = 40)
grid_glm$predicted_response <- exp(grid_glm$predicted)  # log link -> mu
```

`nutrigon_hull` *Design hull as a closed polygon for plotting*

Description

Returns the convex hull of the design as a closed polygon data frame, suitable for `ggplot2::geom_polygon()`.

Usage

```
nutrigon_hull(fit)
```

Arguments

`fit`	A `"nutrigon_fit"` object (from `nutrigon_fit()`).

Value

A data frame with columns named after the original `x_col` and `y_col`, with the first row repeated at the end to close the polygon.

See Also

`nutrigon_grid()`, `nutrigon_region_polygon()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
nutrigon_hull(fit_lifespan)
```

---

nutrigon_matrix *Angular trade-off matrix for N surfaces, with joint bootstrap uncertainty*

Description

Fits N response surfaces, locates each optimum, and returns the N x N matrix of pairwise angular separations with percentile bootstrap intervals. Each surface is bootstrapped independently (the surfaces are assumed to come from independent samples, e.g. different traits measured on different units or different environments), and in replicate b the b-th bootstrap optimum of every surface is used in all pairs in which that surface appears. Each replicate therefore yields a complete, internally consistent matrix, so the dependence among entries (each surface contributes to N - 1 of them) is preserved, and any summary of the matrix can be given bootstrap uncertainty. Optionally, the stability of an average-linkage hierarchical clustering is assessed in this way.

Usage

```
nutrigon_matrix(
  surfaces,
  B = 200,
  resample = c("case", "stratified"),
  conf_level = 0.95,
  cluster_k = 2,
  n_cloud = 500,
  n_reps_cloud = 5,
  seed = NULL
)
```

Arguments

surfaces	A named list. Each element is a list with data, x_col, y_col, z_col and, optionally, model_type (default "lm"), family, design_id_col, peak (default TRUE), re_formula, re_group_cols and extra_args (as in nutrigon_fit()).
B	Number of bootstrap replicates (default 200).
resample	"case" (default) or "stratified" (within design points; needs design_id_col in every surface).
conf_level	Confidence level of the percentile intervals.
cluster_k	If not NULL (default 2), assess clustering stability with the dendrogram cut into cluster_k groups.
n_cloud, n_reps_cloud	Monte Carlo fallback settings.
seed	Optional random seed (set locally).

Details

With two nutrients, every optimum is described by one polar angle and the matrix is one-dimensional (theta_ij = |theta_i - theta_j|): the ordering of the optimum angles (optima) carries all of its information, and ordination methods such as multidimensional scaling add no further dimension.

Value

An object of class "nutrigon_matrix", a list with:

Theta N x N matrix of point estimates (degrees).

lower, upper N x N matrices of interval limits.

optima Data frame with each surface's optimum location, polar angle, bootstrap interval for the angle, and whether the Monte Carlo fallback was used.

boot_theta B x N matrix of bootstrap polar angles.

pairwise Long-format data frame of the N(N-1)/2 entries.

clustering If requested: the point-estimate hclust object, the bootstrap cophenetic correlations, a co-clustering frequency matrix, and per-surface stability (proportion of replicates in which a surface clusters with the members of its point-estimate cluster).

See Also

nutrigon_compare()

Examples

```
data(lee_drosophila)
surfaces <- list(
  lifespan = list(data = lee_drosophila, x_col = "protein_eaten",
                  y_col = "carb_eaten", z_col = "lifespan"),
  eggs     = list(data = lee_drosophila, x_col = "protein_eaten",
                  y_col = "carb_eaten", z_col = "dailyeggs"),
```

```
  lifetime = list(data = lee_drosophila, x_col = "protein_eaten",
                  y_col = "carb_eaten", z_col = "lifetimeegg"))
m <- nutrigon_matrix(surfaces, B = 50, seed = 1, cluster_k = NULL)
m
```

`nutrigon_region_polygon` *Peak (or valley) region as a convex hull polygon for plotting*

Description

Returns the convex hull of the top- (or bottom-) quantile region stored in a `"nutrigon_fit"` object, as a closed polygon data frame suitable for `ggplot2::geom_polygon()`.

Usage

```
nutrigon_region_polygon(fit, quiet = FALSE)
```

Arguments

`fit`	A `"nutrigon_fit"` object (from `nutrigon_fit()`).
`quiet`	Suppress the link-scale reminder message for GLM fits (default `FALSE`).

Value

A data frame with the same column names as `fit$region`, closed into a polygon.

GLM fits

the region stored in `fit$region` (and therefore returned here) was extracted on the **link scale** for `model_type = "glm"` fits, consistent with `nutrigon_grid()` and the optimum-finding logic in `nutrigon_fit()`. The (x, y) coordinates returned are always in the original nutrient units regardless of model type – it is only the *criterion* used to select which points fall inside the region (top/bottom quantile of the link-scale prediction) that differs. A reminder message is printed for GLM fits; pass `quiet = TRUE` to suppress it.

See Also

`nutrigon_grid()`, `nutrigon_hull()`

Examples

```
data(lee_drosophila)
fit_lifespan <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten",
                             "lifespan", design_id_col = "treatment")
nutrigon_region_polygon(fit_lifespan)
```

`nutrigon_snr` *Signal-to-noise ratio of a fitted surface*

Description

The standard deviation of the fitted values over the observed data divided by the residual standard deviation, equivalent to `sqrt(R^2 / (1 - R^2))` up to degrees-of-freedom corrections. Pass the SNRs of your two surfaces to `nutrigon_build_grid(snr = ...)` so that the validation grid reflects how flat your surfaces are relative to their noise; this matters most for small angles.

Usage

```
nutrigon_snr(fit)
```

Arguments

`fit`	A `"nutrigon_fit"` object from `nutrigon_fit()`.

Details

Defined for Gaussian fits (`model_type = "lm"`, and population-level predictions for `"lmer"`). For `"glm"` fits there is no single residual standard deviation on the link scale, so an error is returned; the validation grids assume Gaussian surfaces.

Value

A single positive number.

See Also

`nutrigon_build_grid()`

Examples

```
data(lee_drosophila)
fit <- nutrigon_fit(lee_drosophila, "protein_eaten", "carb_eaten", "lifespan")
nutrigon_snr(fit)
```

`nutrigon_validation_grid` *Pre-built generic validation grid (seed = 2025)*

Description

A validation grid simulated on a generic 5-rail design (angular support 5.71-84.29 degrees) with `seed = 2025`, so that `nutrigon_check()` can be tried immediately. It was built with nutrigon 0.1.0 and does not match any particular experiment: for applied results, build a design-matched grid with `nutrigon_build_grid(design = your_data, snr = ...)`. `nutrigon_check()` warns when this grid's angular support differs from that of the result.

Usage

```
nutrigon_validation_grid
```

Format

A data frame with 1000 rows and columns:

max_elongation Maximum curvature elongation ratio across the two simulated surfaces in that configuration.

min_edge_distance Minimum distance (degrees) from either surface's optimum to the angular boundary of the design.

angle_diff True angular separation (degrees) between the two simulated optima.

coverage Ground-truth empirical bootstrap CI coverage for that configuration, from repeated simulation.

n_valid_sims Number of simulated datasets that produced a valid (non-error) bootstrap result for that configuration, out of the `n_sim_per_point` attempted.

Build settings are stored in the `"build_settings"` attribute: `n_grid_points = 1000`, `n_sim_per_point = 20`, `B_boot = 100`, `n_rails = 5`, `seed = 2025`.

See Also

`nutrigon_check()`, `nutrigon_build_grid()`

`zanco_survival` *Zanco et al. (2025) Drosophila viability data*

Description

Egg-to-adult viability (proportion of eggs eclosing to adulthood) for *Drosophila melanogaster*, measured per vial, across three temperature regimes and two locally-adapted populations, on a geometric array of protein:carbohydrate diets. Used in the package vignette alongside zanco_wingsize as a worked, unpaired comparison (viability is measured per vial; wing size is measured per individual fly, so the two traits do not share a sampling unit).

Usage

```
zanco_survival
```

Format

A data frame with 1495 rows and 15 columns:

Code Integer vial identifier.

Block Integer experimental block.

Temperature Temperature regime: `"18fluc"`, `"25fluc"`, or `"28fluc"` (fluctuating around 18, 25, or 28 degrees C).

Population `"Melbourne"` or `"Townsville"`.

Food Total food concentration.

Ratio Nominal protein:carbohydrate ratio, as a string, e.g. `"(1:8)"`.

Replicate Replicate identifier within design point (character).

Ratio_Num Nominal protein:carbohydrate ratio, as a number (protein fraction).

Prot, Carb Protein and carbohydrate concentration of the diet.

Total_Eggs Number of eggs placed in the vial (out of which adults may eclose).

Total_Adults Number of adults that eclosed.

dead_adults Number of eggs that did not eclose as adults.

Prop_viable Proportion of eggs surviving to adulthood in that vial (the trait of interest).

design_id Design-point identifier (Ratio x Food combination); used for within-design-point bootstrap resampling.

Details

The single constant-25 degree design point (a single design point with no angular design, present in the raw data) has already been excluded, since a quadratic surface cannot be fit to it.

Source

Zanco, B., Morimoto, J., Cockerell, F., Mirth, C. & Sgro, C.M. (2025). Nutritional optima for life-history traits vary with temperature and across locally-adapted populations. *Journal of Insect Physiology*, 163, 104815. doi:10.1016/j.jinsphys.2025.104815

See Also

zanco_wingsize, zanco_tradeoff_matrix

`zanco_tradeoff_matrix` *Pre-computed pairwise angular trade-off matrix for the Zanco example*

Description

The full N x N pairwise angular trade-off matrix across all 12 surfaces fitted to zanco_survival and zanco_wingsize (viability and wing size, each at 3 temperatures x 2 populations), together with the fitted surfaces themselves. The 66 pairwise comparisons are unpaired bootstrap comparisons (`nutrigon_compare(..., paired = FALSE, B = 200)`) and are expensive to compute (several minutes), so this object is shipped pre-built rather than recomputed at package load or in the vignette.

Usage

```
zanco_tradeoff_matrix
```

Format

A list with components:

Theta A 12x12 numeric matrix of angular separations (degrees), named by surface (`"<trait>_<temperature>_<populatio` e.g. `"viability_18fluc_Melbourne"`).

ci_lower, ci_upper Matching 12x12 matrices of 95\ bounds.

pairwise The same information as a flat data frame with one row per pair (66 rows), including `surface_1`, `surface_2`, `trait_1`, `trait_2`, `temperature_1`, `temperature_2`, `population_1`, `population_2`, `theta_deg`, `ci_lower`, `ci_upper`.

surfaces A named list of the 12 `"nutrigon_fit"` objects underlying the matrix, for use with the plot helpers (`nutrigon_grid()`, `nutrigon_hull()`, `nutrigon_region_polygon()`).

build_settings List recording `B` (bootstrap resamples), `temps`, and `pops` used to build the matrix.

See Also

zanco_survival, zanco_wingsize, `nutrigon_compare()`

`zanco_wingsize` *Zanco et al. (2025) Drosophila wing size data*

Description

Wing size for individual female *Drosophila melanogaster*, measured across the same three temperature regimes and two populations as zanco_survival, on the same geometric array of protein:carbohydrate diets. Wing size is measured per individual fly (unlike viability, which is measured per vial), so comparisons between the two traits are unpaired.

Usage

```
zanco_wingsize
```

Format

A data frame with 2073 rows and 9 columns:

Food Total food concentration.

Ratio Nominal protein:carbohydrate ratio, as a string, e.g. `"(1:8)"`.

Replicate Integer replicate identifier within design point.

Prot, Carb Protein and carbohydrate concentration of the diet.

Population `"Melbourne"` or `"Townsville"`.

Temperature Temperature regime: `"18fluc"`, `"25fluc"`, or `"28fluc"`.

WingSize Wing size of the individual fly (the trait of interest).

design_id Design-point identifier (Ratio x Food combination); used for within-design-point bootstrap resampling.

Details

The single constant-25 degree design point has already been excluded, since a quadratic surface cannot be fit to it.

Source

Zanco, B., Morimoto, J., Cockerell, F., Mirth, C. & Sgro, C.M. (2025). Nutritional optima for life-history traits vary with temperature and across locally-adapted populations. *Journal of Insect Physiology*, 163, 104815. doi:10.1016/j.jinsphys.2025.104815

See Also

zanco_survival, zanco_tradeoff_matrix

Index

∗ **datasets**
lee_drosophila, 2
nutrigon_validation_grid, 24
zanco_survival, 25
zanco_tradeoff_matrix, 26
zanco_wingsize, 27

lee_drosophila, 2

nutrigon_build_grid, 3
nutrigon_build_grid(), *6, 7*, *16*, *23, 24*
nutrigon_check, 6
nutrigon_check(), *3*, *5*, *11*, *13–15*, *24*
nutrigon_coefs, 8
nutrigon_coefs(), *17*
nutrigon_compare, 9
nutrigon_compare(), *5–7*, *14–17*, *21*, *26*
nutrigon_curvature, 12
nutrigon_elongation, 13
nutrigon_elongation(), *6*, *12, 13*, *17*
nutrigon_equivalence, 14
nutrigon_equivalence(), *7*, *11*
nutrigon_fit, 15
nutrigon_fit(), *8*, *10–14*, *18, 19*, *21–23*
nutrigon_grid, 18
nutrigon_grid(), *13*, *20*, *22*, *26*
nutrigon_hull, 19
nutrigon_hull(), *19*, *22*, *26*
nutrigon_matrix, 20
nutrigon_region_polygon, 22
nutrigon_region_polygon(), *19, 20*, *26*
nutrigon_snr, 23
nutrigon_snr(), *4, 5*
nutrigon_validation_grid, 24

saveRDS(), *5*
stats::glm(), *10*

zanco_survival, 25, *26, 27*
zanco_tradeoff_matrix, 26, *26, 27*
zanco_wingsize, *25, 26*, 27