Package ‘fb4package’

May 7, 2026

Type Package

Title 'Fish Bioenergetics 4.0' Model Implementation with High-Performance 'TMB' Backend

Version 2.0.0

Description An implementation of the 'Fish Bioenergetics 4.0' framework described in Deslauriers et al. (2017) <doi:10.1080/03632415.2017.1377558>. Provides automated parameter optimization, multi-prey diet modeling, and comprehensive energy budget simulations for fisheries research and aquaculture applications. An optional 'TMB' (Template Model Builder) backend delivers 10-50x speedup in maximum likelihood estimation while maintaining full backward compatibility. Includes species-specific parameter databases and tools for modeling fish growth, consumption, and metabolism under varying environmental conditions.

URL `https://hansttito.github.io/fb4package/`,
`https://github.com/HansTtito/fb4package`

BugReports `https://github.com/HansTtito/fb4package/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports graphics, grDevices, stats, tools, utils, Rcpp, TMB (>= 1.7.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, microbenchmark, future, furrr, spelling

LinkingTo TMB, RcppEigen

Config/testthat/edition 3

RoxygenNote 7.3.2

Language en-US

NeedsCompilation yes

Author Hans Ttito [aut, cre]

Maintainer Hans Ttito <`kvttitos@gmail.com`>

Repository CRAN

Date/Publication 2026-05-07 20:40:13 UTC

Contents

`accumulate_validations` *Accumulate multiple validation results*

Description

Combines multiple validation results into a single result, aggregating errors, warnings, and info messages.

Usage

```
accumulate_validations(..., level = "combined")
```

Arguments

`...`	Validation result objects to combine
`level`	Overall validation level for the combined result

Value

An object of class `fb4_validation` (see `validation_result`) representing the combined state of all inputs. `valid` is `TRUE` only if all supplied results are valid. `errors` and `warnings` are the concatenation of those fields across all inputs.

Examples

```
r1 <- validation_result(valid = TRUE)
r2 <- validation_result(valid = FALSE, errors = "value out of range")
accumulate_validations(r1, r2)
```

`analysis-core` *Core Analysis Functions for FB4 Results*

Description

Core functions for extracting and accessing results from FB4 simulations. These functions provide a unified interface to access results regardless of the fitting method used (`"direct"`, `"optim"`, `"binary_search"`, `"mle"`, `"bootstrap"`, `"hierarchical"`). Exported functions include `is.fb4_result`, `get_consumption_uncertainty`, `get_efficiency_uncertainty`, `get_individual_results`, `get_population_results`
and `get_energy_budget_uncertainty`.

Value

No return value; this page documents the core analysis functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`analysis-extraction` *Basic Analysis and Extraction Functions for FB4 Results*

Description

Functions for basic analysis and extraction of FB4 simulation results. These functions build on the core extraction functions to provide meaningful biological interpretations and statistical summaries. Exported functions include `analyze_growth_patterns`, `analyze_energy_budget`, `analyze_feeding_performance`, and `create_result_summary`.

Value

No return value; this page documents the result extraction and summary functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`analysis-nutritional` *Nutritional Analysis Functions for FB4 Results*

Description

Specialized functions for nutritional analysis of FB4 simulation results. Includes N:P ratio analysis (`calculate_np_ratios`, `compare_with_redfield`), nutrient retention efficiency calculations (`calculate_nutrient_efficiencies`), stoichiometric balance assessment (`calculate_stoichiometric_balance`), body composition analysis (`analyze_composition_by_size`, `analyze_composition_changes`), diet quality assessment (`assess_diet_quality`), and an integrated wrapper (`comprehensive_nutritional_analysis`).

Value

No return value; this page documents the nutritional analysis functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`analysis-sensitivity` *Sensitivity and Comparative Analysis Functions for FB4 Results*

Description

Functions for sensitivity analysis, comparative studies, and population-level analysis of FB4 simulation results. Includes individual comparisons (`compare_individuals`), population variation decomposition (`analyze_population_variation`), temperature-feeding sensitivity analysis (`analyze_growth_temperature`
and multi-scenario comparisons (`compare_scenarios`).

Value

No return value; this page documents the sensitivity analysis functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`analyze_composition_by_size`
Analyze body composition by size range

Description

Analyzes body composition across a range of fish sizes to understand allometric relationships and size-dependent changes.

Usage

```
analyze_composition_by_size(
  weight_range = c(1, 500),
  n_points = 50,
  processed_composition_params
)
```

Arguments

`weight_range`	Weight range to analyze (2-element vector), default c(1, 500)
`n_points`	Number of points to analyze, default 50
`processed_composition_params`	Processed composition parameters

Value

A `data.frame` with `n_points` rows and ten columns: `Weight` (g), `Water_g`, `Protein_g`, `Ash_g`, `Fat_g` (all in g), `Water_fraction`, `Protein_fraction`, `Ash_fraction`, `Fat_fraction` (dimensionless fractions of total wet weight), and `Energy_density` (J/g wet weight).

Examples

```
comp_params <- process_composition_params(list())
comp_analysis <- analyze_composition_by_size(c(1, 500), 20, comp_params)
plot(comp_analysis$Weight, comp_analysis$Energy_density)
```

`analyze_composition_changes`

Analyze composition changes with growth

Description

Analyzes how body composition changes as fish grow during a simulation. Useful for understanding ontogenetic changes in energy density and macronutrient allocation.

Usage

```
analyze_composition_changes(result, processed_composition_params)
```

Arguments

`result`	FB4 result object with daily output
`processed_composition_params`	Processed composition parameters

Value

A `data.frame` with one row per simulation day and thirteen columns: `Weight` (g), `Water_g`, `Protein_g`, `Ash_g`, `Fat_g` (g), `Water_fraction`, `Protein_fraction`, `Ash_fraction`, `Fat_fraction` (dimensionless), `Energy_density` (J/g), `Day` (integer), `Energy_density_change` (J/g/day; `NA` on day 1), `Fat_fraction_change`, and `Protein_fraction_change` (change per day; `NA` on day 1). Stops with an error if `result` has no `daily_output`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
```

```
    ),
    diet_data = list(
      proportions = data.frame(Day = 1:30, Prey1 = 1.0),
      energies    = data.frame(Day = 1:30, Prey1 = 5000),
      prey_names  = "Prey1"
    ),
    simulation_settings = list(initial_weight = 100, duration = 30)
  )
  bio$species_params$predator$ED_ini <- 5000
  bio$species_params$predator$ED_end <- 5500
  result      <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
  comp_params <- process_composition_params(list())
  df <- analyze_composition_changes(result, comp_params)
```

`analyze_energy_budget` *Analyze energy budget from FB4 results*

Description

Analyzes energy budget components from FB4 simulation results. Calculates proportional allocation to different processes with uncertainty propagation when available.

Usage

```
analyze_energy_budget(result, individual_id = NULL, confidence_level = 0.95)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population/single individual)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with four elements:

energy_components The list returned by `get_energy_budget_uncertainty`, containing six component sub-lists each with `estimate`, `se`, `ci_lower`, and `ci_upper`.

proportions Named list of proportional allocations (`prop_respiration`, `prop_egestion`, `prop_excretion`, `prop_sda`, `prop_net`), each a sub-list with `estimate`, `se`, `ci_lower`, and `ci_upper`. `NULL` when consumption energy is zero or unavailable.

summary_metrics Named list with `gross_growth_efficiency`, `metabolic_scope`, and `assimilation_efficiency` sub-lists (each `estimate` + `se`). `NULL` when proportions are unavailable.

balance_check Named list with `consumption_energy`, `total_allocated`, `balance_error`, and `relative_error` (all numeric) to verify mass-balance closure.

Plus the context scalars `method`, `has_uncertainty`, and `individual_id`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
budget <- analyze_energy_budget(result)
```

`analyze_feeding_performance`

Analyze feeding performance from FB4 results

Description

Analyzes feeding-related metrics including consumption rates, feeding efficiency, and p_value estimates with uncertainty.

Usage

```
analyze_feeding_performance(
  result,
  individual_id = NULL,
  confidence_level = 0.95
)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population/single individual)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with at minimum `method` (character), `has_uncertainty` (logical), and `individual_id`. Additional elements present when the relevant data are available:

total_consumption The list returned by `get_consumption_uncertainty` (`estimate`, `se`, `ci_lower`, `ci_upper`, plus context scalars).

daily_consumption Sub-list (`estimate`, `se`, `ci_lower`, `ci_upper`) for the daily consumption rate (g/day).

specific_consumption Sub-list (same four slots) for the specific consumption rate (g consumption / g fish / day).

p_value Structure depends on method: for hierarchical it contains `population_mean`, `population_se`, `population_sd`, and `n_individuals`; for single-individual methods it contains `estimate`, `se`, `ci_lower`, and `ci_upper`.

feeding_efficiency Sub-list (`estimate`, `se`, `ci_lower`, `ci_upper`) for the ratio of total growth to total consumption (dimensionless).

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
feeding <- analyze_feeding_performance(result)
```

`analyze_growth_patterns` *Analyze growth patterns from FB4 results*

Description

Extracts and analyzes growth patterns from FB4 simulation results. Calculates growth rates, efficiency metrics, and provides uncertainty estimates when available.

Usage

```
analyze_growth_patterns(result, individual_id = NULL, confidence_level = 0.95)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population/single individual)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with at minimum `method` (character), `has_uncertainty` (logical), `individual_id`, and `initial_weight` (numeric, g). The following growth metrics are included as sub-lists each with `estimate`, `se`, `ci_lower`, and `ci_upper`: `final_weight` (g), `total_growth` (g), and `relative_growth` (%). When simulation duration is available, `daily_growth_rate` (g/day) and `specific_growth_rate` (%/day) are appended. For fitted methods a `p_value` sub-list (`estimate`, `se`) is also included; for hierarchical population-level calls, `n_individuals` (integer) is added.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
growth <- analyze_growth_patterns(result)
```

`analyze_growth_temperature_sensitivity`

Analyze growth rate sensitivity to temperature and feeding levels

Description

Analyzes how growth rates respond to different temperature and feeding level combinations. Uses p-values (proportion of maximum consumption capacity) to simulate feeding scenarios from survival levels (p ~0.2) to maximum capacity (p = 1.0). Parallelized for efficiency.

Usage

```
analyze_growth_temperature_sensitivity(
  bio_obj,
  temperatures = seq(8, 18, by = 2),
  p_values = seq(0.3, 1, by = 0.1),
  simulation_days = 365,
  oxycal = 13560,
  parallel = FALSE,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

`bio_obj`	Bioenergetic object containing species parameters and environmental data
`temperatures`	Vector of temperatures to test in °C (default: 8-18°C by 2°C steps)
`p_values`	Vector of p-values representing feeding levels as proportion of Cmax (default: 0.3-1.0)
`simulation_days`	Number of days to simulate (default: 365)
`oxycal`	Oxycalorific coefficient in J/g O2 (default: 13560)
`parallel`	Use parallel processing (default: FALSE)
`n_cores`	Number of cores for parallel processing (default: detectCores() - 1)
`verbose`	Show progress information (default: TRUE)

Value

A data frame with one row per temperature-p_value combination and columns:

`temperature` Temperature tested (°C).

`p_value` Feeding level (proportion of Cmax) tested.

`growth_rate` Specific growth rate (percent per day).

`final_weight` Predicted final weight (g).

`total_consumption` Total consumption over the simulation period (g).

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
results <- analyze_growth_temperature_sensitivity(
  bio,
  temperatures     = c(2, 14),
  p_values         = c(0.4, 0.7),
  simulation_days  = 30,
  verbose          = FALSE
)
```

`analyze_population_variation` *Analyze population variation in hierarchical models*

Description

Analyzes the magnitude and sources of variation in hierarchical models. Decomposes total variation into between-individual and within-individual components.

Usage

```
analyze_population_variation(result, include_covariates = TRUE)
```

Arguments

`result`	FB4 result object from hierarchical method
`include_covariates`	Include covariate effects in analysis

Value

A named list with at minimum three elements:

n_individuals Integer. Number of individuals in the analysis.

population_parameters Named list with sub-lists `mu_p`, `sigma_p`, and `sigma_obs`, each containing `estimate`, `se`, `ci_lower`, and `ci_upper`.

variance_decomposition Named list (present when total variance is positive) with `between_individual_variance`, `within_individual_variance`, `total_variance`, `between_individual_prop`, `within_individual_prop`, and `intraclass_correlation`.

When individual outcome data are available, `outcome_variation` is appended (sub-lists `consumption` and optionally `growth`, each with `variance`, `cv`, and `range`). When covariate effects are present and `include_covariates = TRUE`, `covariate_effects` is also appended. Stops with an error if `result` was not produced by the hierarchical method.

Examples

```
# Population variation requires a hierarchical run; shown here for illustration
# result <- run_fb4(bio, strategy = "hierarchical", ...)
# pv <- analyze_population_variation(result)
```

`assess_diet_quality` *Assess nutritional quality of diet*

Description

Assesses the nutritional quality of the diet based on energy density, macronutrient composition, and digestibility of prey items.

Usage

```
assess_diet_quality(diet_data, prey_energies, prey_digestibility = NULL)
```

Arguments

`diet_data`	Diet composition data from FB4 simulation
`prey_energies`	Energy densities of prey items
`prey_digestibility`	Digestibility coefficients of prey items

Value

A named list whose elements depend on whether the diet is time-varying or static. Always present: `mean_energy_density` (numeric, J/g), `energy_density_sd` (numeric), and `energy_density_range` (numeric vector of length 2). For time-varying diets, `daily_energy_density` (numeric vector) is also included. When `prey_digestibility` is supplied, the list additionally contains `mean_digestibility`, `digestibility_sd`, and (for time-varying diets) `daily_digestibility`. A `diversity` sub-list with `mean_shannon` and `shannon_sd` (and `daily_shannon` for time-varying diets) is always appended.

Examples

```
diet  <- list(proportions = data.frame(Day = 1:5,
                                       Prey1 = c(0.6, 0.6, 0.7, 0.5, 0.5),
                                       Prey2 = c(0.4, 0.4, 0.3, 0.5, 0.5)))
prey_e <- data.frame(Day = 1:5, Prey1 = 5000, Prey2 = 4500)
assess_diet_quality(diet, prey_e)
```

`basic-validators` *Basic Validation Functions for FB4*

Description

Basic validation and utility functions built on top of the core validators in `core-validators`. They cover four common needs: scalar numeric validation (`check_numeric_value`), fundamental model-parameter feasibility (`validate_basic_params`), a safe fallback empty-composition constructor (`create_empty_composition`), and time-series structural checks (`validate_time_series_data`).

Value

No return value; this page documents the basic validation functions module. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`Bioenergetic` *Constructor for Bioenergetic Objects*

Description

Creates a Bioenergetic class object that encapsulates all components of the fish bioenergetic model for streamlined simulation management.

Usage

```
Bioenergetic(
  species_params,
  species_info = NULL,
  environmental_data = NULL,
  diet_data = NULL,
  reproduction_data = NULL,
  model_options = list(),
  simulation_settings = list()
)
```

Arguments

`species_params`	List with species parameters organized by categories
`species_info`	List with species identification information
`environmental_data`	List with environmental data (temperature, etc.)
`diet_data`	List with diet and prey energy data
`reproduction_data`	List with reproduction parameters (optional)
`model_options`	List with model configuration options
`simulation_settings`	List with simulation configuration

Details

The Bioenergetic object serves as a comprehensive container for all bioenergetic model components.

Required Components:

species_params Parameter sets for consumption, respiration, etc.

species_info Species identification with scientific_name or common_name

Optional Components:

environmental_data Temperature and other environmental variables

diet_data Diet composition and prey energy densities

model_options Sub-model toggles and advanced settings

simulation_settings Initial conditions and duration

Value

An object of class `"Bioenergetic"`: a named list with eight elements: `species_info`, `species_params`, `environmental_data`, `diet_data`, `reproduction_data`, `model_options`, `simulation_settings`, and `fitted` (logical, `FALSE` until a simulation is run). A `results` element is appended by `set_environment`, `set_diet`, and `run_fb4` when they reset or populate the object.

Examples

```
# Create species parameters
params <- list(
  consumption = list(CEQ = 2, CA = 0.303, CB = -0.275, CQ = 3, CTO = 15, CTM = 25),
  respiration = list(REQ = 1, RA = 0.0548, RB = -0.299, RQ = 2, RTO = 5, RTM = 25)
)

# Create species info
species_info <- list(
  scientific_name = "Salmo salar",
  common_name = "Atlantic salmon",
  life_stage = "juvenile"
)

# Create bioenergetic object
bio_obj <- Bioenergetic(
  species_params = params,
  species_info = species_info,
  simulation_settings = list(initial_weight = 10, duration = 365)
)
```

`bioenergetic-classes` *S3 Classes for FB4 Bioenergetic Model*

Description

S3 class system for the Fish Bioenergetics 4.0 model, providing structured data containers and configuration methods for bioenergetic simulations. The central class `"Bioenergetic"` is created by `Bioenergetic` and configured via `set_environment`, `set_diet`, and `set_simulation_settings`. Utility functions `is.Bioenergetic`, `get_parameter_value`, and `set_parameter_value` support inspection and modification of the object.

Value

No return value; this page documents the S3 class definitions for the bioenergetic model. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`bioenergetic-methods` *Methods for FB4 Bioenergetic Model*

Description

S3 `print` and `summary` methods for the two main classes of the FB4 package: `"Bioenergetic"` (the model configuration object) and `"fb4_result"` (the simulation output). `print` methods produce a concise one-screen overview; `summary` methods extend that with detailed per-method diagnostics (optimisation convergence, MLE statistics, bootstrap percentiles, or hierarchical population parameters).

Value

No return value; this page documents the S3 methods for bioenergetic objects. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`body-composition` *Body Composition Functions for FB4 Model*

Description

Functions for estimating and updating fish body composition (water, protein, ash, fat) and energy density from total wet weight. Composition is estimated via the allometric regressions of Breck (2014):

$$\log_{10}(\text{Component}) = \alpha + \beta \cdot \log_{10}(H_2O)$$

Fat is obtained by subtraction and bounded to biologically plausible limits. Energy density is computed as a weighted sum of fat and protein energy contents (J/g).

Value

No return value; this page documents the body composition functions module. See individual function documentation for return values.

References

Breck, J.E. (2014). Body composition in fishes: body size matters. *Aquaculture*, 433, 40–49. doi:10.1016/j.aquaculture.2014.05.049

`calculate_body_composition` *Calculate complete body composition (Mid-level - Main function)*

Description

Main function that calculates all body composition components and energy density from weight and processed parameters

Usage

```
calculate_body_composition(weight, processed_composition_params)
```

Arguments

`weight`	Total wet weight (g)
`processed_composition_params`	List with processed composition parameters

Value

A named list with 13 elements describing the body composition:

total_weight Numeric. Total wet weight (g), equal to `weight`.

water_g Numeric. Water content (g).

protein_g Numeric. Protein content (g), estimated from water via Breck (2014) regression.

ash_g Numeric. Ash content (g), estimated from water via Breck (2014) regression.

fat_g Numeric. Fat content (g), calculated by subtraction and bounded to `[0, max_fat_fraction * weight]`.

water_fraction Numeric. Water as a fraction of total weight.

protein_fraction Numeric. Protein as a fraction of total weight.

ash_fraction Numeric. Ash as a fraction of total weight.

fat_fraction Numeric. Fat as a fraction of total weight.

energy_density Numeric. Energy density (J/g wet weight).

total_energy Numeric. Total body energy (J).

total_fraction Numeric. Sum of all four fractions; should be close to 1.

balanced Logical. `TRUE` if `total_fraction` is within 0.05 of 1.

Returns `create_empty_composition()` when `weight <= 0`.

Examples

```
params <- list(water_fraction = 0.72, fat_energy = 36450,
               protein_energy = 17990, max_fat_fraction = 0.30)
calculate_body_composition(weight = 100,
                           processed_composition_params = params)
```

`calculate_consumption` *Calculate daily consumption (Mid-level - Main function)*

Description

Main consumption calculation function called from simulation loop

Usage

```
calculate_consumption(
  temperature,
  weight,
  p_value,
  processed_consumption_params,
  method = "rate"
)
```

Arguments

`temperature`	Water temperature (°C)
`weight`	Fish weight (g)
`p_value`	Proportion of maximum consumption (0-5)
`processed_consumption_params`	List with processed consumption parameters
`method`	Calculation method ("maximum", "rate", "specific")

Value

A non-negative numeric scalar giving the daily specific consumption rate in g prey per g fish per day. Returns `0` when the temperature-dependence factor is zero (e.g. temperature $\geq$ CTM in equation 2). The value depends on `method`: `"rate"` (default) scales by `p_value` ($C_{\max} \cdot p \cdot F(T)$); `"maximum"` and `"specific"` return the unscaled value ($C_{\max} \cdot F(T)$).

Examples

```
# CEQ 1: simple exponential temperature dependence
params <- list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06)
calculate_consumption(temperature = 15, weight = 100, p_value = 0.5,
                      processed_consumption_params = params)
```

`calculate_contaminant_accumulation`

Calculate contaminant accumulation (Mid-level - Main function)

Description

Calculates daily contaminant dynamics (uptake, elimination, body burden) for a fish using one of three bioaccumulation models (CONTEQ 1-3).

Usage

```
calculate_contaminant_accumulation(
  respiration_o2,
  consumption,
  weight,
  temperature,
  current_concentration,
  processed_contaminant_params
)
```

Arguments

`respiration_o2`	Respiration in g O2/g/day
`consumption`	Vector of consumption by prey type (g/day)
`weight`	Fish weight (g)
`temperature`	Water temperature (deg C)
`current_concentration`	Current concentration in predator (ug/g)
`processed_contaminant_params`	List with processed contaminant parameters

Value

A named list with at least six elements (all numeric scalars unless noted):

clearance Daily elimination of contaminant (ug/day).

uptake Total daily uptake from food (ug/day); for CONTEQ 3 this is the sum of water and food uptake.

new_burden Body burden at end of day (ug); floored at 0.

new_concentration Whole-body concentration (ug/g wet weight); floored at 0.

weight Fish weight (g), as supplied.

model_used Integer. CONTEQ equation used (1, 2, or 3).

CONTEQ 3 (Arnot & Gobas 2004) appends two additional elements: `uptake_water` (ug/day from water) and `uptake_food` (ug/day from food).

Experimental

Contaminant modelling is an **experimental feature** under active development. This function can be called directly to compute daily bioaccumulation for a single time step, but it is **not yet integrated** into the main ‘run_fb4()‘ simulation loop. Full integration (automatic contaminant tracking across all simulation days, inclusion in ‘fb4_result‘ objects, and TMB backend support) is planned for a future release. The API may change.

Examples

```
# CONTEQ 1: food uptake only, no elimination
params <- list(
  CONTEQ = 1,
  prey_concentrations = c(0.05, 0.08),
  transfer_efficiency = c(0.8, 0.8)
)
calculate_contaminant_accumulation(
  respiration_o2 = 0.02, consumption = c(2.0, 1.0),
  weight = 100, temperature = 15,
  current_concentration = 0.1,
  processed_contaminant_params = params
)
```

`calculate_egestion` *Calculate daily egestion (Mid-level - Main function)*

Description

Main egestion calculation function called from simulation loop

Usage

```
calculate_egestion(
  consumption,
  temperature,
  p_value,
  processed_egestion_params
)
```

Arguments

`consumption`	Consumption (J/g)
`temperature`	Water temperature (deg C)
`p_value`	Proportion of maximum consumption (p_value)
`processed_egestion_params`	List with processed egestion parameters

Value

A non-negative numeric scalar giving the daily egestion rate in J per g fish per day. Returns `0` when `consumption` is zero. The equation used depends on `processed_egestion_params$EGEQ` (1 = constant fraction; 2-3 = temperature- and ration-dependent; 4 = temperature-dependent only). The result is always capped at `consumption` (egestion cannot exceed intake).

Examples

```
# EGEQ 1: constant fraction of consumption
params <- list(EGEQ = 1, FA = 0.16)
calculate_egestion(consumption = 5.0, temperature = 15, p_value = 0.5,
                   processed_egestion_params = params)
```

`calculate_excretion` *Calculate daily excretion (Mid-level - Main function)*

Description

Main excretion calculation function called from simulation loop

Usage

```
calculate_excretion(
  consumption,
  egestion,
  temperature,
  p_value,
  processed_excretion_params
)
```

Arguments

`consumption`	Consumption (J/g)
`egestion`	Egestion (J/g)
`temperature`	Water temperature (deg C)
`p_value`	Proportion of maximum consumption (p_value)
`processed_excretion_params`	List with processed excretion parameters

Value

A non-negative numeric scalar giving the daily excretion rate in J per g fish per day. Returns `0` when `consumption` is zero. The equation used depends on `processed_excretion_params$EXEQ` (1 = constant fraction of assimilated energy; 2-3 = temperature- and ration-dependent; 4 = temperature-dependent only).

Examples

```
# EXEQ 1: constant fraction of assimilated energy
params <- list(EXEQ = 1, UA = 0.1)
calculate_excretion(consumption = 5.0, egestion = 0.8, temperature = 15,
                    p_value = 0.5, processed_excretion_params = params)
```

`calculate_final_weight_fb4` *Calculate final weight using FB4 equations (Mid-level)*

Description

Main function for calculating final weight from energy balance

Usage

```
calculate_final_weight_fb4(
  initial_weight,
  net_energy,
  spawn_energy = 0,
  processed_predator_params,
  day = 1
)
```

Arguments

`initial_weight`	Initial weight (g)
`net_energy`	Net available energy (J)
`spawn_energy`	Energy lost to reproduction (J)
`processed_predator_params`	Processed predator parameters
`day`	Current day

Value

A named list with three elements:

final_weight Numeric scalar. Fish weight (g) at end of day, after accounting for net energy and reproductive losses. Minimum value is `0.01` g.

final_energy_density Numeric scalar. Energy density (J/g) corresponding to `final_weight`.

weight_change Numeric scalar. Change in weight (g) during the day (`final_weight - initial_weight`); negative values indicate weight loss.

Examples

```
# PREDEDEQ 3: power function, positive net energy (weight gain)
params <- list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1)
calculate_final_weight_fb4(initial_weight = 100, net_energy = 500,
                           processed_predator_params = params)
```

`calculate_mortality_reproduction`

Calculate daily mortality and reproduction (Mid-level - Main function)

Description

Calculates daily mortality probability and reproductive energy loss (natural mortality, fishing mortality, predation mortality, starvation, and weight-dependent survival) for a single time step.

Usage

```
calculate_mortality_reproduction(
  current_weight,
  temperature,
  day_of_year,
  processed_mortality_params,
  initial_weight = NULL
)
```

Arguments

`current_weight`	Current fish weight (g)
`temperature`	Water temperature (deg C)
`day_of_year`	Day of year (1-365)
`processed_mortality_params`	List with processed mortality parameters
`initial_weight`	Initial weight for relative calculations (optional)

Value

A named list with five elements:

mortality Named list with seven numeric scalars (all daily rates, 0–1): `survival_rate`, `combined_mortality`, `natural_mortality`, `fishing_mortality`, `predation_mortality`, `weight_effect` (increment to mortality due to low body weight), and `temperature_effect` (increment due to thermal stress).

reproduction `NULL` if no spawn pattern is defined in `processed_mortality_params`; otherwise a named list with `weight_loss` (g), `energy_loss` (J), `spawn_fraction` (0–1), and `remaining_weight` (g).

day_of_year Integer. Day of year, as supplied.

current_weight Numeric. Fish weight (g), as supplied.

temperature Numeric. Water temperature (°C), as supplied.

Experimental

Mortality rate modelling is an **experimental feature** under active development. This function can be called directly to compute daily mortality probability for a single time step, but **mortality rates are not yet integrated** into the main ‘run_fb4()‘ simulation loop. Full integration (automatic daily mortality application, population survival tracking, and result reporting) is planned for a future release. The API may change.

Note: **spawning energy loss** (reproductive cost) *is* already integrated into ‘run_fb4()‘ and applies automatically when ‘reproduction_data‘ is supplied to the ‘Bioenergetic‘ object.

Examples

```
params <- list(
  base_mortality      = 0.001,
  natural_mortality   = 0.001,
  fishing_mortality   = 0.0005,
  predation_mortality = 0.0002,
  weight_threshold    = 10,
  starvation_factor   = 2,
  optimal_temp        = 18,
  thermal_tolerance   = 8,
  stress_factor       = 1.5,
  spawn_pattern       = NULL
)
calculate_mortality_reproduction(current_weight = 100, temperature = 15,
                                 day_of_year = 180,
                                 processed_mortality_params = params)
```

`calculate_np_ratios` *Calculate N:P ratios for all processes*

Description

Calculates molar and mass N:P ratios for consumption, growth, excretion and egestion. Useful for understanding nutritional ecology and stoichiometric balance.

Usage

```
calculate_np_ratios(nitrogen_fluxes, phosphorus_fluxes, ratio_type = "mass")
```

Arguments

`nitrogen_fluxes`	List result from calculate_nutrient_balance (nitrogen component)
`phosphorus_fluxes`	List result from calculate_nutrient_balance (phosphorus component)
`ratio_type`	Type of ratio ("mass" or "molar"), default "mass"

Value

A named list with three elements:

ratios Named numeric vector of length 4 giving the N:P ratio for each process (`consumed`, `growth`, `excretion`, `egestion`). Values may be `Inf` when phosphorus is zero and `NaN` when both nutrients are zero.

ratio_type Character. The ratio type as supplied (`"mass"` or `"molar"`).

redfield_ratio Numeric. Reference Redfield ratio: 7.2 for mass ratios and 16 for molar ratios.

Examples

```
nitrogen   <- list(consumed = 10, growth = 3, excretion = 5, egestion = 2)
phosphorus <- list(consumed = 1.5, growth = 0.5, excretion = 0.6, egestion = 0.4)
np_ratios  <- calculate_np_ratios(nitrogen, phosphorus)
np_ratios$ratios
```

`calculate_nutrient_balance` *Calculate nutrient balance (Mid-level - Main function)*

Description

Calculates daily nitrogen and phosphorus fluxes (ingestion, retention, excretion) for a fish using prey and predator elemental concentrations.

Usage

```
calculate_nutrient_balance(consumption, weight_gain, processed_nutrient_params)
```

Arguments

`consumption`	Vector of consumption by prey type (g/day)
`weight_gain`	Predator weight gain (g/day)
`processed_nutrient_params`	List with processed nutrient parameters

Value

A named list with three elements:

nitrogen Named list with six numeric scalars describing daily nitrogen fluxes (g N/day): `consumed`, `assimilated`, `growth`, `excretion`, `egestion`, and `assimilation_efficiency` (dimensionless fraction, 0–1).

phosphorus Same structure as `nitrogen` but for phosphorus (g P/day).

weight_gain Numeric scalar. Predator weight gain (g/day), as supplied.

Experimental

Nutrient regeneration modelling is an **experimental feature** under active development. This function can be called directly to compute daily N and P fluxes for a single time step, but it is **not yet integrated** into the main ‘run_fb4()‘ simulation loop. Full integration (automatic daily nutrient tracking, inclusion in ‘fb4_result‘ objects, and TMB backend support) is planned for a future release. The API may change.

Examples

```
params <- list(
  prey_n_concentrations     = c(0.025, 0.030),
  prey_p_concentrations     = c(0.004, 0.005),
  predator_n_concentration  = 0.030,
  predator_p_concentration  = 0.004,
  n_assimilation_efficiency = c(0.80, 0.80),
  p_assimilation_efficiency = c(0.60, 0.60)
)
calculate_nutrient_balance(consumption = c(2.0, 1.0),
                           weight_gain = 0.5,
                           processed_nutrient_params = params)
```

`calculate_nutrient_efficiencies`

Calculate nutrient retention efficiencies

Description

Calculates assimilation and retention efficiencies for nitrogen and phosphorus. These metrics are important for understanding nutrient use efficiency.

Usage

```
calculate_nutrient_efficiencies(nitrogen_fluxes, phosphorus_fluxes)
```

Arguments

`nitrogen_fluxes`	List result from calculate_nutrient_balance (nitrogen component)
`phosphorus_fluxes`	List result from calculate_nutrient_balance (phosphorus component)

Value

A named list with four elements:

nitrogen Named list with four numeric scalars: `assimilation_efficiency` (fraction consumed that is assimilated), `retention_efficiency` (fraction consumed retained in growth), `excretion_rate` (fraction consumed lost via excretion), and `growth_efficiency` (fraction assimilated retained in growth).

phosphorus Same structure as `nitrogen` but for phosphorus.

relative_n_retention Numeric. Ratio of nitrogen to phosphorus retention efficiency; `NA` when phosphorus retention is zero.

relative_n_excretion Numeric. Ratio of nitrogen to phosphorus excretion rate; `NA` when phosphorus excretion rate is zero.

Examples

```
nitrogen   <- list(consumed = 10, assimilated = 8, growth = 3, excretion = 5,
                   egestion = 2, assimilation_efficiency = 0.8)
phosphorus <- list(consumed = 1.5, assimilated = 1.1, growth = 0.5,
                   excretion = 0.6, egestion = 0.4, assimilation_efficiency = 0.73)
calculate_nutrient_efficiencies(nitrogen, phosphorus)
```

`calculate_predator_energy_density`

Calculate predator energy density (Mid-level - Main function)

Description

Main energy density calculation function called from simulation loop

Usage

```
calculate_predator_energy_density(weight, day = 1, processed_predator_params)
```

Arguments

`weight`	Fish weight (g)
`day`	Simulation day (for equation 1)
`processed_predator_params`	List with processed predator parameters

Value

A numeric scalar giving the predator energy density in J per g fish, calculated according to `processed_predator_params$PR` (1 = interpolated daily data; 2 = piecewise linear by weight; 3 = power function of weight).

Examples

```
# PREDEDEQ 3: power function of weight
params <- list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1)
calculate_predator_energy_density(weight = 100, processed_predator_params = params)
```

`calculate_respiration` *Calculate daily respiration (Mid-level - Main function)*

Description

Main respiration calculation function called from simulation loop

Usage

```
calculate_respiration(temperature, weight, processed_respiration_params)
```

Arguments

`temperature`	Water temperature (°C)
`weight`	Fish weight (g)
`processed_respiration_params`	List with processed respiration parameters (includes activity params)

Value

A positive numeric scalar giving the daily specific respiration rate in g O_2 per g fish per day. Returns `0.000001` as a minimum safety floor when the result is non-finite or non-positive (e.g. at or above the lethal temperature `RTM`). The value accounts for both the temperature-dependence function (REQ 1 or 2) and the activity multiplier.

Examples

```
# REQ 2: Kitchell et al. (1977) temperature dependence
params <- list(REQ = 2, RA = 0.0033, RB = -0.227,
               RTM = 30, RTO = 18, RX = 0.5, ACT = 1.5)
calculate_respiration(temperature = 15, weight = 100,
                      processed_respiration_params = params)
```

`calculate_stoichiometric_balance` *Calculate stoichiometric balance*

Description

Analyzes nutritional limitations based on N:P ratios and determines which nutrient is limiting growth.

Usage

```
calculate_stoichiometric_balance(nutrient_balance)
```

Arguments

`nutrient_balance`
Complete list result from calculate_nutrient_balance with efficiencies and ratios

Value

A named list with ten elements:

nutrient_limitation Character. Overall assessment: `"N-limited"`, `"P-limited"`, or `"Undetermined"`.

limiting_nutrient Character. The identified limiting nutrient (`"nitrogen"`, `"phosphorus"`, or `"unknown"`).

excess_nutrient Character. The nutrient in relative excess.

excess_factor Numeric. Fold-excess of the non-limiting nutrient relative to the Redfield ratio; 1 when undetermined.

limiting_efficiency Numeric. Retention efficiency of the limiting nutrient; `NA` when undetermined.

consumption_np_ratio Numeric. Observed N:P ratio of consumed food.

redfield_ratio Numeric. Reference Redfield ratio used.

np_deviation Numeric. Difference between observed and Redfield N:P ratio.

efficiencies List from `calculate_nutrient_efficiencies`.

np_ratios List from `calculate_np_ratios`.

Examples

```
nb <- list(
  nitrogen   = list(consumed = 10, assimilated = 8, growth = 3, excretion = 5,
                    egestion = 2, assimilation_efficiency = 0.8),
  phosphorus = list(consumed = 1.5, assimilated = 1.1, growth = 0.5,
                    excretion = 0.6, egestion = 0.4, assimilation_efficiency = 0.73)
)
calculate_stoichiometric_balance(nb)
```

`check_numeric_value` *Check Numeric Value*

Description

Fast validation of numeric values with basic range checking. Simplified utility function for common validation needs.

Usage

```
check_numeric_value(value, name, min_val = -Inf, max_val = Inf)
```

Arguments

`value`	Value to validate
`name`	Parameter name for error messages
`min_val`	Minimum allowed value (default -Inf)
`max_val`	Maximum allowed value (default Inf)

Details

Performs essential validations:

- Not NULL
- Numeric type
- Finite values (no NA, NaN, Inf)
- Within specified range

Value

The original `value` (numeric, same length as input), returned unchanged when all checks pass. Throws an error if `value` is `NULL`, non-numeric, contains non-finite elements (`NA`, `NaN`, `Inf`), or falls outside `[min_val, max_val]`.

Examples

```
check_numeric_value(5, "weight")
try(check_numeric_value(-1, "weight", min_val = 0))
try(check_numeric_value(NA, "weight"))
```

`compare_individuals` *Compare individuals from hierarchical models*

Description

Compares performance metrics between individuals in hierarchical models. Provides statistical summaries and identifies outliers.

Usage

```
compare_individuals(result, metrics = "all", confidence_level = 0.95)
```

Arguments

`result`	FB4 result object from hierarchical method
`metrics`	Vector of metrics to compare ("consumption", "growth", "efficiency", "all")
`confidence_level`	Confidence level for comparisons (default 0.95)

Value

A named list with at minimum three context elements: `n_individuals` (integer), `metrics_compared` (character vector), and `confidence_level` (numeric). Depending on `metrics`, the following sublists are appended; each is produced by an internal summary helper and contains `metric_name`, `n_valid`, `mean`, `sd`, `min`, `max`, `median`, `cv`, `range`, `outliers`, and `performance`: `consumption`, `efficiency`, and `p_value`. The `growth` element (when requested) is itself a list with two such summaries (`total_growth` and `relative_growth`). A `rankings` `data.frame` (one row per individual; columns for per-metric ranks, `composite_rank`, and `overall_rank`) is always appended. Stops with an error if `result` was not produced by the hierarchical method.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
mr_data <- data.frame(
  individual_id  = paste0("fish_", 1:5),
  initial_weight = c(10, 12, 11, 13, 9),
  final_weight   = c(80, 95, 85, 100, 70)
)
result_hier <- run_fb4(bio, strategy = "hierarchical", backend = "tmb",
                       fit_to = "Weight", observed_weights = mr_data,
                       verbose = FALSE)
comparison <- compare_individuals(result_hier)
```

`compare_scenarios` *Compare multiple FB4 results*

Description

Compares multiple FB4 simulation results across different scenarios, parameters, or methods. Useful for comparing alternative models or experimental conditions.

Usage

```
compare_scenarios(
  result_list,
  metrics = c("consumption", "growth", "efficiency"),
  confidence_level = 0.95
)
```

Arguments

`result_list`	Named list of FB4 result objects
`metrics`	Vector of metrics to compare
`confidence_level`	Confidence level for comparisons

Value

A named list with five elements:

n_scenarios Integer. Number of scenarios compared.

scenario_names Character vector of scenario names.

metrics_compared Character vector of metrics requested.

confidence_level Numeric. Confidence level as supplied.

scenario_data `data.frame` with one row per scenario. Always contains `scenario`, `method`, `backend`, and `converged`. Additional `*_est` and `*_se` columns are appended for each requested metric (`consumption`, `growth`, `efficiency`, `p_value`).

When at least two scenarios provide uncertainty estimates, `statistical_tests` is appended (list of pairwise test results). `best_performers` (named list of scenario names with highest estimated value per metric) is always appended.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
```

```
bio$species_params$predator$ED_end <- 5500
r1 <- run_fb4(bio, strategy = "direct", p_value = 0.4, verbose = FALSE)
r2 <- run_fb4(bio, strategy = "direct", p_value = 0.7, verbose = FALSE)
comparison <- compare_scenarios(list(low_p = r1, high_p = r2))
```

`compare_with_redfield` *Compare N:P ratios with Redfield ratios*

Description

Compares calculated N:P ratios with the classical Redfield ratio. Useful for understanding deviations from typical oceanic proportions.

Usage

```
compare_with_redfield(np_ratios)
```

Arguments

`np_ratios`	List result from calculate_np_ratios

Value

A `data.frame` with one row per process and six columns: `Process` (character), `NP_Ratio` (numeric), `Redfield_Ratio` (numeric), `Difference` (numeric; observed minus Redfield), `Relative_Difference` (numeric; % deviation from Redfield), and `Interpretation` (character; one of `"N-rich relative to P"`, `"N-poor relative to P"`, `"No P available"`, or `"No flux"`).

Examples

```
nitrogen   <- list(consumed = 10, growth = 3, excretion = 5, egestion = 2)
phosphorus <- list(consumed = 1.5, growth = 0.5, excretion = 0.6, egestion = 0.4)
np  <- calculate_np_ratios(nitrogen, phosphorus)
compare_with_redfield(np)
```

`comprehensive_nutritional_analysis` *Comprehensive nutritional analysis*

Description

Performs a comprehensive nutritional analysis combining all nutritional metrics including N:P ratios, nutrient efficiencies, body composition, and diet quality assessment.

Usage

```
comprehensive_nutritional_analysis(
  result,
  nutrient_balance = NULL,
  composition_params = NULL,
  diet_quality_data = NULL
)
```

Arguments

`result`	FB4 result object
`nutrient_balance`	Nutrient balance results (if available)
`composition_params`	Body composition parameters (if available)
`diet_quality_data`	Diet quality data (if available)

Value

A named list with at minimum two elements: `model_info` (list with `method` and `has_daily_output`) and `energy_budget` (from `analyze_energy_budget`). When optional inputs are provided, the following elements are appended:

np_ratios, redfield_comparison, nutrient_efficiencies, stoichiometric_balance Added when `nutrient_balance` is supplied.

initial_composition, final_composition, composition_changes Added when `composition_params` is supplied and both initial and final weights are available in `result`.

diet_quality Added when `diet_quality_data` is supplied.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
```

```
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
analysis <- comprehensive_nutritional_analysis(result)
```

`consumption-functions` *Consumption Functions for FB4 Model*

Description

Functions implementing the four consumption temperature-dependence equations (CEQ 1–4) and the allometric maximum-consumption function used in FB4. Consumption is modelled as:

$$C = C_{\max} \cdot p \cdot F(T), \quad C_{\max} = CA \cdot W^{CB}$$

where p is the proportion of maximum consumption (P-value), $F(T)$ is a temperature-dependence function, W is body mass (g), and CA, CB are species-specific intercept and slope coefficients.

CEQ 1 — simple Q10 exponential: $F(T) = e^{CQ \cdot T}$

CEQ 2 — Kitchell et al. (1977): $F(T) = V^{CX} \cdot e^{CX(1-V)}$, where $V = (CTM - T)/(CTM - CTO)$

CEQ 3 — Thornton and Lessem (1978): two-part sigmoid using CQ, CTO, CTL, CTM, $CK1$, $CK4$

CEQ 4 — polynomial: $F(T) = e^{CQ \cdot T + CK1 \cdot T^2 + CK4 \cdot T^3}$

Value

No return value; this page documents the consumption functions module. See individual function documentation for return values.

References

Kitchell, J.F., Stewart, D.J. and Weininger, D. (1977). Applications of a bioenergetics model to yellow perch and walleye. *Journal of the Fisheries Research Board of Canada*, 34(10), 1922–1935. doi:10.1139/f77258

Thornton, K.W. and Lessem, A.S. (1978). A temperature algorithm for modifying biological rates. *Transactions of the American Fisheries Society*, 107(2), 284–287.

Hartman, K.J. and Hayward, R.S. (2007). Bioenergetics. In C.S. Guy and M.L. Brown (eds.), *Analysis and Interpretation of Freshwater Fisheries Data*. American Fisheries Society, Bethesda, MD.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`contaminant-accumulation` *Contaminant Accumulation Functions for FB4 Model*

Description

Experimental functions for modelling daily contaminant (e.g. methylmercury, PCBs) dynamics in fish using three bioaccumulation models (CONTEQ 1–3).

CONTEQ 1 — food uptake only, no elimination: $\text{Burden}_{t+1} = \text{Burden}_t + \sum_i C_i \cdot [\text{prey}]_i \cdot \text{AE}_i$

CONTEQ 2 — food uptake with temperature- and weight-dependent elimination (Trudel and Rasmussen 1997): $K_x = \exp(0.066T - 0.2 \ln W - 6.56)$

CONTEQ 3 — Arnot and Gobas (2004): uptake from both water (via gill transfer) and food, elimination proportional to respiration.

Value

No return value; this page documents the contaminant accumulation functions module. See individual function documentation for return values.

References

Arnot, J.A. and Gobas, F.A.P.C. (2004). A food web bioaccumulation model for organic chemicals in aquatic ecosystems. *Environmental Toxicology and Chemistry*, 23(10), 2343–2355. doi:10.1897/03438

Trudel, M. and Rasmussen, J.B. (1997). Modeling the elimination of mercury by fish. *Environmental Science and Technology*, 31(6), 1716–1722. doi:10.1021/es960609t

---

`core-validators` *Core Validation Functions for FB4*

Description

Atomic validation functions that provide the foundation for all other validation operations. These functions handle the most basic validation patterns used throughout the FB4 system: numeric range checks, structural requirements (required columns, minimum rows), and domain-specific validators for fractions, positive quantities, and temperatures. All validators return standardised `fb4_validation` objects constructed by `validation_result`, which can be aggregated with `accumulate_validations`.

Value

No return value; this page documents the core validation functions module. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`create_empty_composition` *Create empty composition for invalid inputs (Utility)*

Description

Create empty composition for invalid inputs (Utility)

Usage

```
create_empty_composition()
```

Value

A named list with 13 numeric/logical elements, all set to zero or `FALSE`: `total_weight`, `water_g`, `protein_g`, `ash_g`, `fat_g`, `water_fraction`, `protein_fraction`, `ash_fraction`, `fat_fraction`, `energy_density`, `total_energy`, `total_fraction` (all `0`), and `balanced` (`FALSE`). Used as a safe fallback when fish weight is zero or negative.

Examples

```
create_empty_composition()
```

`create_result_summary` *Comprehensive post-simulation analysis summary*

Description

Post-hoc analysis function — takes a finished `fb4_result` object and bundles growth, feeding, and energy-budget analyses into a single list. Useful when you need all major metrics in one call.

This is different from the internal `$summary` slot (built automatically during result construction by `create_unified_summary()`). This function re-derives richer metrics from `daily_output` and supports uncertainty propagation for MLE / bootstrap / hierarchical results.

Usage

```
create_result_summary(result, individual_id = NULL, confidence_level = 0.95)
```

Arguments

`result`	An `fb4_result` object returned by `run_fb4()`
`individual_id`	For hierarchical models: individual ID to extract (`NULL` returns population-level summary)
`confidence_level`	Confidence level for uncertainty intervals, default 0.95

Value

Named list with `model_info`, `growth`, `feeding`, `energy_budget`, and `model_fit` sections

See Also

`analyze_growth_patterns`, `analyze_feeding_performance`, `analyze_energy_budget`

`data-processing` *Data Processing Functions for FB4*

Description

Functions for preparing, validating, and transforming the temporal and parameter data required by the FB4 simulation engine. The main entry point is `prepare_simulation_data`, which orchestrates species-parameter processing (via `process_species_parameters`) and temporal-data processing (via `process_bioenergetic_data`). Ancillary functions handle time-series interpolation (`interpolate_time_series`), diet normalisation, reproduction scheduling, and TMB-format transformations for statistical fitting strategies.

Value

No return value; this page documents the data processing functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`data-validators` *Data Validation Functions for FB4*

Description

Data validation functions built on top of the core validators in `core-validators`. Covers diet-energy consistency (`validate_diet_consistency`), individual mark-recapture data (`validate_individual_data`), processed temporal arrays (`validate_temporal_data`), the complete simulation data structure (`validate_complete_simulation_data`), and equation parameter requirements (`validate_equation_params`).

Value

No return value; this page documents the data validation functions module. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`egestion-excretion` *Egestion and Excretion Functions for FB4 Model*

Description

Functions implementing four egestion models (EGEQ 1–4) and four excretion models (EXEQ 1–4). These represent losses of consumed energy as feces (F) and nitrogenous wastes (U) respectively.

Egestion models:

EGEQ 1 — constant fraction: $F = FA \cdot C$

EGEQ 2 — Elliott (1976), temperature- and feeding-dependent: $F = FA \cdot T^{FB} \cdot e^{FG \cdot p} \cdot C$

EGEQ 3 — Stewart et al. (1983), includes indigestible fraction: modified form of EGEQ 2 accounting for indigestible prey material.

EGEQ 4 — Elliott (1976), temperature-dependent only: $F = FA \cdot T^{FB} \cdot C$

Excretion models:

EXEQ 1 — constant fraction of assimilated energy: $U = UA \cdot (C - F)$

EXEQ 2–3 — Elliott (1976), temperature- and feeding-dependent: $U = UA \cdot T^{UB} \cdot e^{UG \cdot p} \cdot (C - F)$

EXEQ 4 — temperature-dependent only.

Value

No return value; this page documents the egestion and excretion functions module. See individual function documentation for return values.

References

Elliott, J.M. (1976). Energy losses in the waste products of brown trout (*Salmo trutta* L.). *Journal of Animal Ecology*, 45(2), 561–580.

Stewart, D.J., Weininger, D., Rottiers, D.V. and Edsall, T.A. (1983). An energetics model for lake trout, *Salvelinus namaycush*: application to the Lake Michigan population. *Canadian Journal of Fisheries and Aquatic Sciences*, 40(6), 681–698.

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fb4-analysis-plots` *Analysis Plots for FB4 Results (Uncertainty and Sensitivity)*

Description

Plotting functions for uncertainty analysis and sensitivity analysis. Exported functions include `plot_uncertainty.fb4_result` (dispatches to MLE, bootstrap, and hierarchical sub-functions), `plot_distributions.fb4_result`, `plot_sensitivity.fb4_result`, and `plot_growth_temperature_sensitivity`.

Value

No return value, called for side effects (plots). See individual function documentation for details.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fb4-bioenergetic-plots` *Bioenergetic Object Plots for Setup Validation*

Description

Plotting functions for `Bioenergetic` objects (before running a simulation). These plots help validate model setup by displaying temperature profiles (`plot_bio_temperature`), diet composition over time (`plot_bio_diet`), predator energy density (`plot_bio_energy`), and an integrated readiness dashboard (`plot_bio_dashboard`).

Value

No return value, called for side effects (plots). See individual function documentation for details.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fb4-daily-plots` *Daily Simulation Plots for FB4 Results*

Description

Plotting functions for daily simulation output. These functions work with the `daily_output` data produced by `run_fb4()` and visualise growth (`plot_growth`), consumption (`plot_consumption`), temperature (`plot_temperature`), and energy (`plot_energy`) patterns over time, as well as an integrated dashboard (`plot_dashboard`).

Value

No return value, called for side effects (plots). See individual function documentation for details.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fb4-plot-core` *Core Plotting Functions for FB4 Results*

Description

Core utilities and helper functions for the FB4 visualization system. Provides consistent color schemes (`get_color_scheme`), plot layout helpers (`setup_plot_layout`), graphics-device management (`setup_save_device`, `close_save_device`), annotation utilities (`add_confidence_bands`, `add_plot_annotations`), and data validation helpers (`validate_plot_data`) shared across all plotting modules.

Value

No return value, called for side effects (plots). See individual function documentation for details.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fb4-plots` *FB4 Plotting Functions*

Description

Main plotting functions for FB4 bioenergetic model results. Provides S3 methods (`plot.fb4_result`, `plot.Bioenergetic`) that dispatch to specialised plot types (`"dashboard"`, `"growth"`, `"consumption"`, `"temperature"`, `"energy"`, `"uncertainty"`, `"sensitivity"`).

Value

No return value, called for side effects (plots). See individual function documentation for details.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`FB4-TMB-Shared` *FB4 TMB Shared Functions*

Description

Internal helper functions shared by the TMB-based MLE and hierarchical fitting strategies. Covers TMB objective validation (`validate_tmb_objective`), DLL availability checks (`check_tmb_compilation`), robust multi-start optimization (`run_robust_optimization`), parameter extraction from sdreport summaries (`sdr_pull_est`, `sdr_pull_se`, `sdr_pull_vec`, `sdr_assign_scalars`), and result assembly for basic and hierarchical models (`extract_tmb_results`).

Value

No return value; this page documents shared TMB backend functions. See individual function documentation for return values.

References

Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H. and Bell, B.M. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5), 1–21. doi:10.18637/jss.v070.i05

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`fish4_parameters` *Fish Bioenergetics 4.0 Official Parameters Database*

Description

Comprehensive database containing species-specific bioenergetic parameters for the Fish Bioenergetics 4.0 model. This database includes consumption, respiration, egestion, excretion, and predator energy density parameters for multiple fish species across different life stages.

Usage

```
fish4_parameters
```

Format

A list containing bioenergetic parameters for fish species with the following structure:

species_name List for each species containing:

species_info Basic taxonomic information (scientific name, common name, family, order)

life_stages Named list of life stages (e.g., "juvenile", "adult", "larval") containing:

consumption Consumption parameters (CA, CB, CQ, CTO, CTM, CTL, CK1, CK4, CEQ)

respiration Respiration parameters (RA, RB, RQ, RTO, RTM, RTL, RK1, RK4, RK5, REQ)

activity Activity multipliers (ACT, BACT)

sda Specific Dynamic Action coefficient (SDA)

egestion Egestion parameters (FA, FB, FG, EGEQ)

excretion Excretion parameters (UA, UB, UG, EXEQ)

predator Predator energy density parameters (Alpha1, Beta1, Alpha2, Beta2, Cutoff, ED_data, PREDEDEQ)

source Literature source reference

notes Additional notes about the parameters

sources Vector of literature sources for the species

Details

The database contains parameters for fish species from multiple families including Salmonidae, Percidae, Centrarchidae, Cyprinidae, and others. Each species entry includes one or more life stages with complete or partial parameter sets.

Parameter Categories:

Consumption: Temperature-dependent consumption model parameters

Respiration: Metabolic rate and activity parameters

Egestion: Waste production and defecation parameters

Excretion: Nitrogenous waste excretion parameters

Predator Energy Density: Weight-dependent energy content parameters

Temperature Parameters:

CTO, RTO: Optimum temperature for consumption/respiration

CTM, RTM: Maximum temperature for consumption/respiration

CTL, RTL: Lethal temperature for consumption/respiration

Usage: This database is primarily used with the `Bioenergetic` constructor to create species-specific bioenergetic model objects. Parameters can be extracted using utility functions or accessed directly by species and life stage.

Source

Parameters extracted from `Parameters_official.csv`, the official species database bundled with the Fish Bioenergetics 4.0 ('FB4') Shiny application (<https://github.com/biofish/FishBioenergetics>). Converted to R list format via `generate_fish4_parameters()`. Original parameter values compiled from peer-reviewed literature; see the `source` field within each species entry for individual references.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A., Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

See Also

`Bioenergetic`, `run_fb4`

Examples

```
data(fish4_parameters)
head(names(fish4_parameters))
salmon_info <- fish4_parameters[["Oncorhynchus mykiss"]]
names(salmon_info$life_stages)
juvenile_params <- salmon_info$life_stages$juvenile
names(juvenile_params$consumption)
```

`fish4_parameters_metadata` *Fish Bioenergetics 4.0 Parameters Database Metadata*

Description

Metadata information about the Fish Bioenergetics 4.0 parameters database, including version information, creation details, and structural specifications.

Usage

```
fish4_parameters_metadata
```

Format

A list containing metadata about the fish4_parameters database:

version	Version number of the database
creation_date	Date when the database was generated
source_file	Original CSV file used to generate the database
description	Brief description of the database contents
n_species	Total number of species in the database
n_total_records	Total number of parameter records
families_included	Number of taxonomic families represented
parameter_groups	Vector of parameter category names
required_parameters	List of essential parameters for FB4 simulations
units	Description of parameter units and measurements

Details

This metadata object provides essential information about the structure, content, and requirements of the fish4_parameters database. It includes quality control information and parameter specifications necessary for proper use of the bioenergetic model.

Required Parameters: The metadata specifies which parameters are essential for running Fish Bioenergetics 4.0 simulations:

Consumption: CA, CB, CQ, CTO, CTM, CTL

Respiration: RA, RB, RQ, RTO, RTM, RTL

Units:

Temperature: Degrees Celsius

Energy Density: Joules per gram (J/g)

Weight: Grams (g)

Rates: Proportions or model coefficients

Source

Generated automatically when converting `Parameters_official.csv` (from the Fish Bioenergetics 4.0 Shiny application, <https://github.com/biofish/FishBioenergetics>) into an R list object.

See Also

`fish4_parameters`, `Bioenergetic`

Examples

```
data(fish4_parameters_metadata)
print(fish4_parameters_metadata$description)
print(paste("Species count:", fish4_parameters_metadata$n_species))
```

`get_consumption_uncertainty`

Get consumption results with uncertainty

Description

Extracts consumption results from FB4 simulations with uncertainty propagation when available. Works with all fitting methods.

Usage

```
get_consumption_uncertainty(
  result,
  individual_id = NULL,
  confidence_level = 0.95
)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population mean)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with eight elements:

- **estimate** Numeric. Total consumption estimate (g) for the simulation period; `NA` when unavailable.
- **se** Numeric. Standard error of the estimate; `NA` for methods without uncertainty quantification (e.g. `"direct"`, `"binary_search"`, `"optim"`).
- **ci_lower** Numeric. Lower bound of the confidence interval; `NA` when `se` is unavailable.
- **ci_upper** Numeric. Upper bound of the confidence interval; `NA` when `se` is unavailable.
- **method** Character. Fitting method used (e.g. `"direct"`, `"mle"`, `"hierarchical"`).
- **backend** Character. Computational backend (`"r"` or `"tmb"`).
- **has_uncertainty** Logical. `TRUE` when standard errors and confidence intervals are populated.
- **individual_id** As supplied; the requested individual index, or `NULL` for the population mean.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
consumption <- get_consumption_uncertainty(result)
```

`get_efficiency_uncertainty` *Get efficiency results with uncertainty*

Description

Extracts growth efficiency results from FB4 simulations with uncertainty propagation when available.

Usage

```
get_efficiency_uncertainty(
  result,
  individual_id = NULL,
  confidence_level = 0.95
)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population mean)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with six elements:

gross_growth_efficiency Named sub-list with `estimate`, `se`, `ci_lower`, and `ci_upper` for the gross growth efficiency (dimensionless ratio of growth energy to consumption energy); values are `NA` when unavailable.

metabolic_scope Named sub-list with the same four slots for the metabolic scope (ratio of active to standard metabolism); values are `NA` when unavailable.

method Character. Fitting method used.

backend Character. Computational backend.

has_uncertainty Logical. `TRUE` when SEs and CIs are populated.

individual_id As supplied.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
eff <- get_efficiency_uncertainty(result)
```

`get_energy_budget_uncertainty` *Get energy budget components with uncertainty*

Description

Extracts energy budget components from FB4 simulations with uncertainty propagation when available.

Usage

```
get_energy_budget_uncertainty(
  result,
  individual_id = NULL,
  confidence_level = 0.95
)
```

Arguments

`result`	FB4 result object
`individual_id`	Individual ID for hierarchical models (NULL for population mean)
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list with ten elements: six energy-component sub-lists (`consumption_energy`, `respiration_energy`, `egestion_energy`, `excretion_energy`, `sda_energy`, `net_energy`), each containing `estimate`, `se`, `ci_lower`, and `ci_upper` (all numeric, `NA` when unavailable); plus `method` (character), `backend` (character), `has_uncertainty` (logical), and `individual_id` (as supplied).

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
budget <- get_energy_budget_uncertainty(result)
```

---

get_individual_results

Get individual results from hierarchical models

Description

Extracts all individual-level results from hierarchical FB4 models. Returns a comprehensive summary for each individual.

Usage

```
get_individual_results(result, confidence_level = 0.95)
```

Arguments

`result`	FB4 result object from hierarchical method
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A `data.frame` with one row per individual. Base columns are `individual_id`, `p_estimate`, and `p_se`. When individual uncertainty data are available the frame additionally contains `*_est`, `*_se`, `*_ci_lower`, and `*_ci_upper` columns for `final_weight`, `consumption`, `total_growth`, `relative_growth`, `gross_efficiency`, and `metabolic_scope`. Stops with an error if `result` was not produced by the hierarchical method.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
# Individual results require a hierarchical run; shown here for illustration
# result <- run_fb4(bio, strategy = "hierarchical", ...)
```

```
# df <- get_individual_results(result)
```

`get_parameter_value` *Get Parameter Value from Species Parameters*

Description

Retrieves a specific parameter value from species parameter lists, searching across all parameter categories.

Usage

```
get_parameter_value(params, param)
```

Arguments

`params`	Species parameters list
`param`	Parameter name to retrieve

Value

The value associated with `param` in the first category of `params` where it is found, or `NULL` if `param` is not present in any category. The type of the returned value matches the stored parameter (typically a numeric scalar).

Examples

```
sp <- list(consumption = list(CA = 0.303, CB = -0.275))
get_parameter_value(sp, "CA")
get_parameter_value(sp, "nonexistent")
```

`get_population_results` *Get population results from hierarchical models*

Description

Extracts population-level results from hierarchical FB4 models. Returns means, standard errors, and population parameters.

Usage

```
get_population_results(result, confidence_level = 0.95)
```

Arguments

`result`	FB4 result object from hierarchical method
`confidence_level`	Confidence level for intervals (default 0.95)

Value

A named list containing at minimum ten elements: `mu_p_estimate` and `mu_p_se` (population-mean ration), `sigma_p_estimate` and `sigma_p_se` (among-individual SD), `sigma_obs_estimate` and `sigma_obs_se` (observation SD), `n_individuals` (integer), `log_likelihood`, `aic`, and `bic`. When population uncertainty data are available, additional `mean_*_est`, `mean_*_se`, `mean_*_ci_lower`, and `mean_*_ci_upper` elements are appended for `final_weight`, `consumption`, `total_growth`, `relative_growth`, `gross_efficiency`, and `metabolic_scope`. Confidence-interval elements are also added for `mu_p`, `sigma_p`, and `sigma_obs`. Stops with an error if `result` was not produced by the hierarchical method.

Examples

```
# Population results require a hierarchical run; shown here for illustration
# result <- run_fb4(bio, strategy = "hierarchical", ...)
# pop <- get_population_results(result)
```

`interpolate_time_series`
Interpolate time series with error handling

Description

Interpolate time series with error handling

Usage

```
interpolate_time_series(
  data,
  value_columns,
  target_days,
  method = "linear",
  fill_na_method = "extend",
  validate_input = TRUE
)
```

Arguments

`data`	Data frame with Day column and value columns
`value_columns`	Vector with names of columns to interpolate
`target_days`	Vector of target days
`method`	Interpolation method ("linear", "constant", "spline")
`fill_na_method`	Method to fill missing values ("extend", "zero", "mean")
`validate_input`	Validate input structure, default TRUE

Value

A `data.frame` with one row per element of `target_days`. The first column is `Day` (integer). Subsequent columns correspond to `value_columns`, each containing the interpolated numeric values at the requested days. `NA` values are resolved according to `fill_na_method`: `"extend"` fills with the nearest valid value, `"zero"` replaces with 0, and `"mean"` uses the column mean.

Examples

```
temp_data <- data.frame(Day = c(1, 100, 200, 365),
                        Temperature = c(5, 15, 18, 7))
interpolate_time_series(temp_data, value_columns = "Temperature",
                        target_days = 1:365)
```

`is.Bioenergetic` *Test if Object is Bioenergetic*

Description

Tests whether an object inherits from the Bioenergetic class.

Usage

```
is.Bioenergetic(x)
```

Arguments

`x`	Object to test

Value

A length-1 logical: `TRUE` if `x` inherits from class `"Bioenergetic"`, `FALSE` otherwise.

Examples

```
bio <- Bioenergetic(
  species_params = list(
    consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06)
  ),
  species_info = list(common_name = "Example fish")
)
is.Bioenergetic(bio)
is.Bioenergetic(list())
```

`is.fb4_result` *Test if Object is fb4_result*

Description

Tests whether an object inherits from the fb4_result class.

Usage

```
is.fb4_result(x)
```

Arguments

`x`	Object to test

Value

A length-1 logical: `TRUE` if `x` inherits from class `"fb4_result"`, `FALSE` otherwise.

Examples

```
is.fb4_result(list())
```

`main-validators` *Main Validation Functions for FB4*

Description

Top-level validation functions that orchestrate all lower-level validators. `validate_bioenergetic_for_simulation` checks a `Bioenergetic` object for simulation readiness (structure, species equations, temperature/diet data, initial weight). `validate_fb4_inputs` extends this with strategy- and data-range checks before calling `run_fb4`. `validate_fb4_system` provides a multi-layer diagnostic report (basic, standard, comprehensive) with per-component pass/fail summaries.

Value

No return value; this page documents the main validation functions module. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`mortality-reproduction`

Mortality and Reproduction Functions for FB4 Model

Description

Experimental functions for computing daily fish mortality and reproductive energy costs in the FB4 framework. Mortality is decomposed into natural, fishing, and predation components, with optional adjustments for thermal stress and starvation (weight-dependent). Reproduction is modelled as a seasonal spawn-fraction pattern that removes a fraction of body weight (and its associated energy) on each spawning day.

Note: Mortality rate tracking is not yet integrated into the main `run_fb4()` loop; spawning energy loss is.

Value

No return value; this page documents the mortality and reproduction functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`nutrient-regeneration` *Nutrient Regeneration Functions for FB4 Model*

Description

Experimental functions for computing daily nitrogen (N) and phosphorus (P) fluxes in fish using a mass-balance approach consistent with ecological stoichiometry theory. For each element the daily budget is:

$$\text{Consumed} = \text{Assimilated} + \text{Egested}$$

$$\text{Assimilated} = \text{Growth} + \text{Excreted}$$

Assimilation efficiencies for N and P are species- and prey-specific and can differ from those used for energy.

Value

No return value; this page documents the nutrient regeneration functions module. See individual function documentation for return values.

References

Sterner, R.W. and Elser, J.J. (2002). *Ecological Stoichiometry: The Biology of Elements from Molecules to the Biosphere*. Princeton University Press, Princeton, NJ.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`parameter-processing` *Parameter Processing Functions for FB4*

Description

Functions for validating, transforming, and enriching raw user-supplied species parameters before they are passed to the simulation engine. Each major bioenergetic category (consumption, respiration, egestion, excretion, predator energy density, contaminant, nutrient, mortality, body composition) has a dedicated processor that (i) checks that the required parameters for the chosen equation are present, (ii) computes derived values (e.g., `CX`/`CY`/`CZ` for CEQ 2, gill efficiency for CONTEQ 3), and (iii) fills missing optional parameters with documented defaults. The top-level entry point is `process_species_parameters`.

Value

No return value; this page documents the parameter processing functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`parameter-validators` *Parameter Validation Functions for FB4*

Description

Parameter validation functions built on top of the core validators in `core-validators`. Covers species-equation validation (`validate_species_equations`), predator energy density (`validate_predator_energy_para` contaminant parameters (`validate_contaminant_params`), nutrient concentrations (`validate_nutrient_concentration` and body composition (`validate_body_composition`). A central `EQUATION_REQUIREMENTS` registry stores the required parameters and valid ranges for each bioenergetic equation.

Value

No return value; this page documents the parameter validation functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`plot.Bioenergetic` *Plot Bioenergetic object setup*

Description

Plotting method for Bioenergetic objects to validate setup before simulation.

Usage

```
## S3 method for class 'Bioenergetic'
plot(x, type = "dashboard", save_plot = NULL, ...)
```

Arguments

`x`	Object of class Bioenergetic
`type`	Type of plot: "dashboard", "temperature", "diet", "energy"
`save_plot`	Optional path to save plot
`...`	Additional arguments

Value

Invisibly returns the input object `x`, called for its plotting side-effect.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
plot(bio)
plot(bio, type = "temperature")
```

`plot.fb4_result` *Plot FB4 simulation results*

Description

Main plotting method for fb4_result objects. Automatically detects available data and provides appropriate visualizations.

Usage

```
## S3 method for class 'fb4_result'
plot(x, type = "dashboard", save_plot = NULL, ...)
```

Arguments

`x`	Object of class fb4_result
`type`	Type of plot: "dashboard", "growth", "consumption", "temperature", "energy", "uncertainty", "sensitivity"
`save_plot`	Optional path to save plot (.png or .pdf)
`...`	Additional arguments passed to specific plot functions

Value

Invisibly returns the input object `x`, called for its plotting side-effect.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
plot(result)
plot(result, type = "growth")
```

`plot_distributions.fb4_result` *Plot parameter distributions for bootstrap and hierarchical methods*

Description

Shows distributions of parameters from bootstrap samples or hierarchical individual estimates.

Usage

```
plot_distributions.fb4_result(
  fb4_result,
  color_scheme = "green",
  show_individuals = TRUE
)
```

Arguments

`fb4_result`	FB4 result object with distribution data
`color_scheme`	Color scheme to use, default "green"
`show_individuals`	For hierarchical: show individual estimates, default TRUE

Value

Called for its plotting side-effect. Invisibly returns `NULL`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
set.seed(42)
obs_weights <- rnorm(10, mean = 90, sd = 5)
result_boot <- run_fb4(bio, strategy = "bootstrap", fit_to = "Weight",
                       observed_weights = obs_weights, n_bootstrap = 20,
                       verbose = FALSE)
plot_distributions.fb4_result(result_boot)
```

---

plot_growth_temperature_sensitivity

Plot sensitivity analysis

Description

Creates sensitivity analysis plots for temperature and feeding effects.

Usage

```
plot_growth_temperature_sensitivity(
  sensitivity_data,
  temperatures = seq(5, 20, by = 2),
  feeding_levels = c(0.5, 0.75, 1),
  species = NULL,
  ylim = NULL,
  xlim = NULL,
  colors = "grayscale",
  verbose = FALSE,
  ...
)
```

Arguments

`sensitivity_data`	Data frame from `analyze_growth_temperature_sensitivity`
`temperatures`	Temperature values to test
`feeding_levels`	Feeding levels to test
`species`	Optional species name for plot title
`ylim`	Optional y-axis limits
`xlim`	Optional x-axis limits
`colors`	Color scheme
`verbose`	Show progress messages
`...`	Additional arguments

Value

Called for its plotting side-effect. Invisibly returns `NULL`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
```

```
    species_params     = sp,
    species_info       = info,
    environmental_data = list(
      temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
    ),
    diet_data = list(
      proportions = data.frame(Day = 1:30, Prey1 = 1.0),
      energies    = data.frame(Day = 1:30, Prey1 = 5000),
      prey_names  = "Prey1"
    ),
    simulation_settings = list(initial_weight = 100, duration = 30)
  )
  bio$species_params$predator$ED_ini <- 5000
  bio$species_params$predator$ED_end <- 5500
  sens_data <- analyze_growth_temperature_sensitivity(
    bio_obj         = bio,
    temperatures    = c(10, 14),
    p_values        = c(0.4, 0.7),
    simulation_days = 30,
    verbose         = FALSE
  )
  plot_growth_temperature_sensitivity(sens_data, species = "Chinook")
```

`plot_sensitivity.fb4_result`
Plot temperature sensitivity analysis for a Bioenergetic object

---

Description

Runs `analyze_growth_temperature_sensitivity` and plots the result. Sensitivity analysis requires a `Bioenergetic` object (not an `fb4_result`), because it re-runs the model across a grid of temperatures and p_values. Use `plot(bio_obj, type = "sensitivity")` as the primary interface; this function is the underlying implementation.

Usage

```
plot_sensitivity.fb4_result(
  bio_obj,
  temperatures = seq(4, 20, by = 2),
  p_values = seq(0.3, 1, by = 0.1),
  simulation_days = 365,
  color_scheme = "grayscale",
  add_annotations = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`bio_obj`	Bioenergetic object with species parameters, temperature profile, diet, and simulation settings.
`temperatures`	Numeric vector of absolute temperatures (°C) to test. Default `seq(4, 20, by = 2)`.
`p_values`	Numeric vector of p_values (proportion of Cmax) to evaluate. Must be in (0, 1]. Default `seq(0.3, 1.0, by = 0.1)`.
`simulation_days`	Number of simulation days. Default 365.
`color_scheme`	Color scheme for the plot. Default `"grayscale"`.
`add_annotations`	Add optimal temperature annotations. Default TRUE.
`verbose`	Show analysis progress. Default FALSE.
`...`	Additional arguments passed to `plot_growth_temperature_sensitivity()`.

Value

Called for its plotting side-effect. Invisibly returns `NULL`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
plot_sensitivity.fb4_result(
  bio_obj         = bio,
  temperatures    = c(10, 14),
  p_values        = c(0.4, 0.7),
  simulation_days = 30,
  verbose         = FALSE
)
```

`plot_uncertainty.fb4_result` *Plot parameter uncertainty for probabilistic methods*

Description

Creates plots showing parameter estimates with confidence intervals. Adapts automatically to the method used (MLE, bootstrap, hierarchical).

Usage

```
plot_uncertainty.fb4_result(
  fb4_result,
  parameters = "all",
  color_scheme = "blue",
  add_ci_text = TRUE
)
```

Arguments

`fb4_result`	FB4 result object with uncertainty estimates
`parameters`	Parameters to plot: "p_value", "consumption", "all", default "all"
`color_scheme`	Color scheme to use, default "blue"
`add_ci_text`	Add confidence interval text, default TRUE

Value

Called for its plotting side-effect. Invisibly returns `NULL`.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
```

```
bio$species_params$predator$ED_end <- 5500
set.seed(42)
obs_weights <- rnorm(10, mean = 90, sd = 5)
result_mle <- run_fb4(bio, strategy = "mle", fit_to = "Weight",
                      observed_weights = obs_weights, verbose = FALSE)
plot_uncertainty.fb4_result(result_mle)
```

`predator-energy-density` *Predator Energy Density Functions for FB4 Model*

Description

Functions implementing three predator energy density models (PREDEDEQ 1–3) and the corresponding weight-solving routines used in the FB4 energy balance.

PREDEDEQ 1 — interpolated daily data: energy density supplied as a vector of length $n_{\text{days}} + 1$ (one value per day boundary).

PREDEDEQ 2 — piecewise linear by weight: $ED = \alpha_1 + \beta_1 W$ below the cutoff, $ED = \alpha_2 + \beta_2 W$ above.

PREDEDEQ 3 — power function: $ED = \alpha_1 \cdot W^{\beta_1}$.

Value

No return value; this page documents the predator energy density functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`predict_consumption_bootstrap` *Bootstrap method for consumption uncertainty propagation*

Description

Propagates p-value uncertainty to consumption predictions using parametric bootstrap. Generates multiple samples from the p-value distribution and runs FB4 simulations for each sample. Provides full uncertainty distribution without linearity assumptions. Supports parallel processing for improved performance.

Usage

```
predict_consumption_bootstrap(
  p_mean,
  p_sd,
  bio_obj,
  n_sims = 1000,
  first_day = 1,
  last_day = 365,
  parallel = FALSE,
  n_cores = NULL,
  confidence_level = 0.95,
  verbose = FALSE
)
```

Arguments

`p_mean`	Mean of p-value distribution
`p_sd`	Standard deviation of p-value distribution
`bio_obj`	Bioenergetic object with simulation settings and environmental data
`n_sims`	Number of bootstrap simulations, default 1000
`first_day`	First simulation day, default 1
`last_day`	Last simulation day, default 365
`parallel`	Use parallel processing, default FALSE
`n_cores`	Number of cores for parallel processing (NULL = auto-detect), default NULL
`confidence_level`	Confidence level for intervals, default 0.95
`verbose`	Show progress messages, default FALSE

Details

The bootstrap method: 1. Samples p-values from Normal(p_mean, p_sd) 2. Constrains samples to valid range [0.01, 5.0] 3. Runs FB4 simulation for each p-value sample 4. Summarizes consumption distribution

Parallel processing can significantly reduce computation time for large n_sims. The method handles simulation failures gracefully and reports success rates.

Value

A named list with elements:

`method` Character string `"bootstrap"`.

`consumption_mean` Mean total consumption across bootstrap samples (g).

`consumption_sd` Standard deviation of consumption across samples (g).

`consumption_ci` Numeric vector of length 2 with lower and upper quantile-based confidence interval bounds (g).

`consumption_samples` Numeric vector of all successful consumption estimates from bootstrap samples.

`n_successful` Number of bootstrap iterations that produced valid consumption estimates.

`p_mean` The supplied `p_mean` value.

`p_sd` The supplied `p_sd` value.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
uncertainty_result <- predict_consumption_bootstrap(
  p_mean   = 0.5,
  p_sd     = 0.05,
  bio_obj  = bio,
  n_sims   = 20,
  last_day = 30
)
```

`predict_consumption_delta`

Delta method for consumption uncertainty propagation

Description

Propagates p-value uncertainty to consumption predictions using the delta method. Computes numerical derivatives and applies first-order approximation for uncertainty propagation. Suitable when the relationship between p and consumption is approximately linear.

Usage

```
predict_consumption_delta(
  p_est,
  p_se,
  bio_obj,
  delta_size = 0.001,
  first_day = 1,
  last_day = 365,
  verbose = FALSE
)
```

Arguments

`p_est`	Estimated p-value (feeding level parameter)
`p_se`	Standard error of p-value estimate
`bio_obj`	Bioenergetic object with simulation settings and environmental data
`delta_size`	Small increment for numerical derivative computation, default 0.001
`first_day`	First simulation day, default 1
`last_day`	Last simulation day, default 365
`verbose`	Show progress messages, default FALSE

Details

The delta method uses first-order Taylor series approximation: Var(f(X)) ~ [f'(mu)]^2 * Var(X)

The linearity check verifies that the derivative times delta_size is small relative to the consumption estimate, indicating local linearity.

Value

A named list with elements:

`method` Character string `"delta"`.

`consumption_est` Point estimate of total consumption (g).

`consumption_se` Standard error of consumption estimate (g).

`consumption_ci` Numeric vector of length 2 with lower and upper confidence interval bounds (g).

`derivative` Numerical derivative of consumption with respect to p.

`linearity_check` Logical; `TRUE` if the linearity assumption appears satisfied.

`p_est` The supplied `p_est` value.

`p_se` The supplied `p_se` value.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
uncertainty_result <- predict_consumption_delta(
  p_est    = 0.5,
  p_se     = 0.05,
  bio_obj  = bio,
  last_day = 30
)
```

`prepare_simulation_data`

Prepare all simulation data

Description

Master function that processes and validates ALL data required for FB4 simulation. Combines species parameter processing with temporal data processing.

Usage

```
prepare_simulation_data(
  bio_obj,
  strategy,
  fit_to = NULL,
  fit_value = NULL,
  first_day = 1,
  last_day = NULL,
  validate_inputs = TRUE,
  oxycal = 13560,
  output_format = "simulation",
```

```
  observed_weights = NULL,
  covariates = NULL
)
```

Arguments

`bio_obj`	Bioenergetic object (must be pre-validated)
`strategy`	Strategy to use: "binary_search", "optim", "bootstrap", "mle", "hierarchical"
`fit_to`	Target type for fitting (e.g., "Weight"); optional for direct strategy
`fit_value`	Target value to fit to; optional for direct strategy
`first_day`	First simulation day
`last_day`	Last simulation day
`validate_inputs`	Whether to perform comprehensive validation, default TRUE
`oxycal`	Oxycalorific coefficient (J/g O2), default 13560
`output_format`	Output format: "simulation", "tmb_basic", "tmb_hierarchical"
`observed_weights`	Data frame with columns: individual_id, initial_weight and observed_weight
`covariates`	Optional covariate matrix or data frame or choose a column of individual_data

Value

For `output_format = "simulation"` (default), a named list with seven elements: `species_params` (processed species parameter sub-lists), `temporal_data` (processed temporal arrays), `simulation_settings` (processed settings), `metadata` (processing timestamp, duration, prey species, data sources), `n_days` (integer), `temperatures` (numeric vector), and `initial_weight` (numeric scalar). For `output_format = "tmb_basic"` or `"tmb_hierarchical"`, returns a list formatted for TMB model fitting (structure differs).

Examples

```
# Requires a fully-configured Bioenergetic object; see ?Bioenergetic
# bio <- Bioenergetic(...)
# sim_data <- prepare_simulation_data(bio, strategy = "direct")
```

`print.Bioenergetic` *Print Method for Bioenergetic Objects*

Description

Displays a concise one-page overview of a `Bioenergetic` object, including species identity, initial weight, simulation duration, and the status of each required component (parameters, temperature, diet). Readiness for fitting is reported in the final status line.

Usage

```
## S3 method for class 'Bioenergetic'
print(x, ...)
```

Arguments

`x`	Bioenergetic object
`...`	Additional arguments (not used)

Value

Invisibly returns the input object

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
print(bio)
```

`print.fb4_result` *Print Method for fb4_result Objects*

Description

Displays a concise summary of an `fb4_result` object. The output adapts to the fitting method used: traditional methods (binary search, optim, direct) show weight, growth, consumption, and convergence; `"mle"` shows parameter estimates with confidence intervals and AIC; `"bootstrap"` shows mean/SD estimates and CI; and `"hierarchical"` shows population-level parameters with model fit statistics.

Usage

```
## S3 method for class 'fb4_result'
print(x, ...)
```

Arguments

`x`	fb4_result object
`...`	Additional arguments (not used)

Value

Invisibly returns the input object

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
print(result)
```

`process_bioenergetic_data` *Process Bioenergetic object temporal data for simulation*

Description

Version that processes all temporal data required for FB4 simulation. Includes better error handling and additional data types.

Usage

```
process_bioenergetic_data(bio_obj, first_day, last_day)
```

Arguments

`bio_obj`	Bioenergetic object (must be pre-validated)
`first_day`	First simulation day
`last_day`	Last simulation day

Value

A named list with ten elements containing the temporal arrays interpolated to each simulation day: `temperature` (numeric vector, °C), `diet_proportions` (numeric matrix, rows = days, columns = prey), `prey_energies` (numeric matrix, J/g), `prey_indigestible` (numeric matrix, fractions), `reproduction` (numeric vector, fractions), `duration` (integer, number of days), `prey_names` (character vector), `first_day`, `last_day`, and `target_days` (integer sequence of simulated days).

Examples

```
# Requires a fully-configured Bioenergetic object; see ?Bioenergetic
# bio <- Bioenergetic(...)
# temporal <- process_bioenergetic_data(bio, first_day = 1, last_day = 365)
```

`process_composition_params` *Process body composition parameters*

Description

Process body composition parameters

Usage

```
process_composition_params(composition_params)
```

Arguments

`composition_params`	Raw composition parameters

Value

A list containing all elements of `composition_params` with missing entries filled with defaults: `water_fraction = 0.75`, `fat_energy = 39500` (J/g), `protein_energy = 23600` (J/g), and `max_fat_fraction = 0.25`.

Examples

```
process_composition_params(list(water_fraction = 0.72))
```

`process_consumption_params` *Process consumption parameters*

Description

Process consumption parameters

Usage

```
process_consumption_params(consumption_params)
```

Arguments

`consumption_params`	Raw consumption parameters

Value

A list containing all elements of `consumption_params` plus derived values required by the selected equation: `CX`, `CY`, `CZ` for CEQ 2 (Kitchell et al. 1977); `CG1`, `CG2` for CEQ 3 (Thornton and Lessem 1978). For CEQ 1 and CEQ 4 the input list is returned unchanged after validation.

Examples

```
process_consumption_params(list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06))
```

`process_contaminant_params` *Process contaminant parameters*

Description

Process contaminant parameters

Usage

```
process_contaminant_params(contaminant_params)
```

Arguments

`contaminant_params`	Raw contaminant parameters

Value

A list containing all elements of `contaminant_params`. For CONTEQ 3 (Arnot and Gobas 2004), three additional elements are computed when not already present: `gill_efficiency` (dimensionless), `fish_water_partition` (dimensionless), and `dissolved_fraction` (dimensionless). For CONTEQ 1 and 2 the list is returned after validation unchanged.

Examples

```
process_contaminant_params(list(
  CONTEQ = 1,
  prey_concentrations  = c(0.05, 0.08),
  transfer_efficiency  = c(0.80, 0.80)
))
```

`process_egestion_params`

Process egestion parameters

Description

Process egestion parameters

Usage

```
process_egestion_params(egestion_params)
```

Arguments

`egestion_params`	Raw egestion parameters

Value

A list identical to `egestion_params` after validation. No additional derived values are computed; the function ensures the required parameters for the selected EGEQ are present and valid.

Examples

```
process_egestion_params(list(EGEQ = 1, FA = 0.16))
```

`process_excretion_params` *Process excretion parameters*

Description

Process excretion parameters

Usage

```
process_excretion_params(excretion_params)
```

Arguments

`excretion_params`	Raw excretion parameters

Value

A list identical to `excretion_params` after validation. No additional derived values are computed; the function ensures the required parameters for the selected EXEQ are present and valid.

Examples

```
process_excretion_params(list(EXEQ = 1, UA = 0.10))
```

`process_mortality_params` *Process mortality parameters*

Description

Process mortality parameters

Usage

```
process_mortality_params(mortality_params)
```

Arguments

`mortality_params`	Raw mortality parameters

Value

A list containing all elements of `mortality_params` with missing entries filled with defaults: `base_mortality` = `0.001`, `natural_mortality` (copied from `base_mortality`), `fishing_mortality` = `0`, `predation_mortality` = `0`, `optimal_temp` = `15`, `thermal_tolerance` = `5`, and `stress_factor` = `2`. If a `reproduction` sub-list is present, a `spawn_pattern` numeric vector (length 365) is appended.

Examples

```
process_mortality_params(list(base_mortality = 0.001,
                              natural_mortality = 0.001))
```

---

process_nutrient_params

Process nutrient parameters

Description

Process nutrient parameters

Usage

```
process_nutrient_params(nutrient_params)
```

Arguments

`nutrient_params`	Raw nutrient parameters

Value

A list containing all elements of `nutrient_params`. Default assimilation efficiencies are inserted when absent: `n_assimilation_efficiency` = `0.85` and `p_assimilation_efficiency` = `0.80` (with a warning in each case).

Examples

```
process_nutrient_params(list(
  prey_n_concentrations    = c(0.025, 0.030),
  prey_p_concentrations    = c(0.004, 0.005),
  predator_n_concentration = 0.030,
  predator_p_concentration = 0.004
))
```

`process_predator_params` *Process predator energy density parameters*

Description

Process predator energy density parameters

Usage

```
process_predator_params(predator_params, n_days = NULL)
```

Arguments

`predator_params`	Raw predator parameters
`n_days`	Integer. Number of simulation days, used to build the energy density vector from ‘ED_ini‘/‘ED_end‘ when PREDEDEQ = 1. If ‘NULL‘ the vector is built lazily when the simulation starts.

Value

A list containing all elements of `predator_params`. For PREDEDEQ 1, an `ED_data` numeric vector of length `n_days + 1` is added (or validated if already present). For PREDEDEQ 2 and 3 the list is returned after validation with no additional derived values.

Examples

```
process_predator_params(list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1))
```

`process_respiration_params` *Process respiration parameters*

Description

Process respiration parameters

Usage

```
process_respiration_params(
  respiration_params,
  activity_params = NULL,
  sda_params = NULL
)
```

Arguments

`respiration_params`	Raw respiration parameters
`activity_params`	Activity parameters (required for REQ=1)
`sda_params`	SDA parameters

Value

A list containing all elements of `respiration_params`, the activity parameters (merged from `activity_params`), the SDA coefficient (`SDA`), and — for REQ 2 — the derived values `RX`, `RY`, `RZ` (Kitchell et al. 1977). For REQ 1 no additional derived values are added.

Examples

```
process_respiration_params(
  respiration_params = list(REQ = 2, RA = 0.0033, RB = -0.227,
                            RQ = 0.025, RTM = 30, RTO = 18),
  activity_params = list(ACT = 1.5),
  sda_params = list(SDA = 0.15)
)
```

`process_simulation_settings`

Process simulation settings

Description

Process simulation settings

Usage

```
process_simulation_settings(settings, first_day, last_day, oxycal = 13560)
```

Arguments

`settings`	Raw simulation settings
`first_day`	First simulation day
`last_day`	Last simulation day
`oxycal`	Oxycalorific coefficient (J/g O2), default 13560

Value

A named list with ten elements: `initial_weight` (numeric, g), `duration` (integer, days), `first_day`, `last_day`, `oxycal` (numeric, J/g O2), `output_frequency` (integer), `save_daily_details` (logical), `tolerance` (numeric), `max_iterations` (integer), `step_size` (numeric), and four logical flags: `calculate_composition`, `calculate_contaminants`, `calculate_nutrients`, `track_mortality`.

Examples

```
settings <- list(initial_weight = 100, p_value = 0.5,
                 fit_to = "Weight", fit_value = 200)
process_simulation_settings(settings, first_day = 1, last_day = 365)
```

`process_species_parameters`

Process all species parameters for simulation

Description

Main function that processes and validates all species parameters, calculating derived parameters and preparing them for simulation.

Usage

```
process_species_parameters(species_params, n_days = NULL)
```

Arguments

`species_params`	Raw species parameters from user
`n_days`	Integer. Number of simulation days, used to build the predator energy density vector when ‘ED_ini‘/‘ED_end‘ are supplied (PREDEDEQ = 1). If ‘NULL‘ the vector length is inferred later.

Value

A named list containing one processed sub-list for each category present in `species_params` (e.g. `consumption`, `respiration`, `egestion`, `excretion`, `predator`, and optionally `contaminant`, `nutrient`, `mortality`, `composition`). Each sub-list holds the validated raw parameters plus any derived values required by the chosen equation. A `processing_info` element is always appended containing `processed_at` (POSIXct timestamp), `validation_warnings` (character vector), and `categories_processed` (character vector of processed category names).

Examples

```
sp <- list(
  consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06),
  egestion    = list(EGEQ = 1, FA = 0.16),
  excretion   = list(EXEQ = 1, UA = 0.10),
  predator    = list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1),
  respiration = list(REQ = 2, RA = 0.0033, RB = -0.227,
                     RQ = 0.025, RTM = 30, RTO = 18),
  activity    = list(ACT = 1.5),
  sda         = list(SDA = 0.15)
)
process_species_parameters(sp)
```

`respiration-functions` *Respiration Functions for FB4 Model*

Description

Functions implementing the two respiration temperature-dependence equations (REQ 1–2), activity correction, and conversion of oxygen consumption to energy units. Respiration is modelled as:

$$R = RA \cdot W^{RB} \cdot F(T) \cdot \text{ACT}$$

where RA and RB are species-specific intercept and slope coefficients, W is body mass (g), $F(T)$ is a temperature function, and ACT is an activity multiplier.

REQ 1 — simple Q10 exponential with activity: $F(T) = e^{RQ \cdot T}$; velocity-based activity from Kitchell et al. (1977).

REQ 2 — Kitchell et al. (1977): $F(T) = V^{RX} \cdot e^{RX(1-V)}$, where $V = (RTM - T)/(RTM - RTO)$.

Oxygen consumption is converted to energy using the oxycalorific coefficient (default 13 560 J g^{-1} O_2; Elliott and Davison 1975).

Value

No return value; this page documents the respiration functions module. See individual function documentation for return values.

References

Kitchell, J.F., Stewart, D.J. and Weininger, D. (1977). Applications of a bioenergetics model to yellow perch and walleye. *Journal of the Fisheries Research Board of Canada*, 34(10), 1922–1935. doi:10.1139/f77258

Elliott, J.M. and Davison, W. (1975). Energy equivalents of oxygen consumption in animal energetics. *Oecologia*, 19(3), 195–201. doi:10.1007/BF00345305

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`result-builders-unified` *Result Builders for FB4 Model*

Description

Provides a unified system for assembling `fb4_result` objects from the raw output of any fitting strategy. The main entry point is `build_fb4_result_unified`, which delegates to `create_unified_summary`, `create_method_specific_data`, and `create_unified_fit_info` to populate the three core slots of the result object. Large uncertainty tables for TMB-based strategies are handled by dedicated helpers: `build_tmb_uncertainty`, `build_individual_uncertainty`, and `build_population_uncertainty`.

Value

No return value; this page documents the result builder functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`run-fb4-orchestrator` *FB4 Main Orchestrator*

Description

Top-level entry point for running Fish Bioenergetics 4.0 simulations. `run_fb4` is an S3 generic that dispatches to `run_fb4.Bioenergetic`, which orchestrates input validation, backend selection (pure R or TMB), execution-plan construction, strategy dispatch, and result assembly. Supported strategies are `"direct"`, `"binary_search"`, `"optim"`, `"mle"` (maximum likelihood), `"bootstrap"`, and `"hierarchical"`.

Value

No return value; this page documents the simulation orchestration functions. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`run_fb4` *Run FB4 Simulation*

Description

Generic function that dispatches to `run_fb4.Bioenergetic`. Pass a `Bioenergetic` object as `x`; all other arguments are forwarded to the method.

Default method — throws an informative error when `x` is not a `Bioenergetic` object.

Usage

```
run_fb4(x, ...)

## Default S3 method:
run_fb4(x, ...)
```

Arguments

`x`	A `Bioenergetic` object (see `Bioenergetic`).
`...`	Arguments passed to `run_fb4.Bioenergetic`.

Value

An object of class `fb4_result`. See `run_fb4.Bioenergetic` for full details of the return structure.

No return value. Stops with an informative error message.

`run_fb4.Bioenergetic` *Run FB4 simulation on Bioenergetic object*

Description

S3 method with automatic backend selection and bootstrap estimation. Supports traditional optimization methods, MLE approaches, and new bootstrap estimation for final weight data. This is the main entry point that coordinates all FB4 execution strategies.

Usage

```
## S3 method for class 'Bioenergetic'
run_fb4(
  x,
  fit_to = NULL,
  fit_value = NULL,
  observed_weights = NULL,
  covariates = NULL,
  first_day = 1,
```

```
    last_day = NULL,
    backend = "r",
    strategy = "binary_search",
    oxycal = 13560,
    tolerance = 0.001,
    max_iterations = 25,
    optim_method = "Brent",
    lower = 0.01,
    upper = 5,
    hessian = FALSE,
    verbose = FALSE,
    confidence_level = 0.95,
    estimate_sigma = TRUE,
    compute_profile = FALSE,
    profile_grid_size = 50,
    n_bootstrap = 1000,
    parallel = FALSE,
    n_cores = NULL,
    sample_size = NULL,
    compute_percentiles = TRUE,
    ...
)
```

Arguments

`x`	Bioenergetic object with all model components
`fit_to`	Target type: "Weight", "Consumption", "p_value", "Ration", "Ration_prey"
`fit_value`	Target value for deterministic approach
`observed_weights`	Vector of observed final weights for MLE or bootstrap approaches (optional)
`covariates`	Optional covariate matrix or data frame
`first_day`	First simulation day, default 1
`last_day`	Last simulation day (auto-detected if NULL)
`backend`	Backend selection: "r" (pure R) or "tmb" (C++ via TMB, faster MLE)
`strategy`	Fitting strategy: "binary_search" (default), "direct", "optim", "mle" (maximum likelihood), or "bootstrap" (bootstrap estimation)
`oxycal`	Oxycalorific coefficient (J/g O2), default 13560
`tolerance`	Convergence tolerance for iterative fitting, default 0.001
`max_iterations`	Maximum iterations for binary search, default 25
`optim_method`	If using optim, which method: "Brent", "L-BFGS-B", etc.
`lower`	Lower bound for p_value search (proportion of Cmax), default 0.01
`upper`	Upper bound for p_value search (proportion of Cmax). Biologically, p = 1.0 is maximum ration; values > 1.0 are super-maximal. Default 1.0 for bootstrap, 5.0 for binary_search.

`hessian`	Whether to compute Hessian for standard errors, default FALSE
`verbose`	Whether to show progress messages, default FALSE
`confidence_level`	Confidence level for MLE/bootstrap intervals, default 0.95
`estimate_sigma`	Whether to estimate measurement error in MLE, default TRUE
`compute_profile`	Whether to compute likelihood profile for MLE, default FALSE
`profile_grid_size`	Number of points in profile grid for MLE, default 50
`n_bootstrap`	Number of bootstrap iterations, default 1000
`parallel`	Whether to use parallel processing for bootstrap, default FALSE
`n_cores`	Number of cores for parallel processing (NULL = auto-detect)
`sample_size`	Sample size for each bootstrap iteration (NULL = same as original)
`compute_percentiles`	Whether to compute additional percentiles for bootstrap, default TRUE
`...`	Additional arguments passed to strategy-specific functions (e.g., `store_predicted_weights_boot` for bootstrap)

Value

An object of class `fb4_result`, a named list with four elements:

summary Named list with `method`, `p_estimate`, `final_weight`, `total_consumption`, and method-specific fields (`p_mean`, `p_sd`, confidence intervals for MLE and bootstrap, etc.).

daily_output A `data.frame` with one row per simulation day containing `Day`, `Weight`, `Consumption_energy`, `Respiration`, `Egestion`, `Excretion`, `SDA`, `Net_energy`, `Energy_density`, and related columns.

method_data Method-specific auxiliary data: bootstrap p-value distributions and percentiles, MLE likelihood profile, or hierarchical population parameters, depending on `strategy`.

bioenergetic_object The original `Bioenergetic` object `x` supplied by the caller.

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
```

```
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
result$summary$final_weight
```

`run_fb4_simulation` *Run complete FB4 simulation (Mid-level - Main function)*

Description

Main simulation function that executes the complete FB4 model day by day. Handles different consumption methods and optional daily output. Performs basic validation of critical parameters.

Usage

```
run_fb4_simulation(
  consumption_method,
  processed_simulation_data,
  oxycal = 13560,
  output_daily = TRUE,
  verbose = FALSE
)
```

Arguments

`consumption_method`	List with method type and value
`processed_simulation_data`	Complete processed simulation data from prepare_simulation_data()
`oxycal`	Oxycalorific coefficient (J/g O2), default 13560
`output_daily`	Whether to save daily outputs, default TRUE
`verbose`	Whether to show progress messages, default FALSE

Value

A named list with up to eleven elements:

initial_weight Numeric. Starting fish weight (g).

final_weight Numeric. Fish weight at end of simulation (g); minimum 0.01 g.

weight_change Numeric. Net change in weight (g).

relative_growth Numeric. Relative growth as percentage of initial weight.

total_consumption_g Numeric. Cumulative consumption over all simulated days (g prey).

total_consumption Numeric. Alias for `total_consumption_g` (retained for backward compatibility).

simulation_days Integer. Number of days actually simulated; may be less than the full duration if mortality occurs.

method List. The `consumption_method` supplied by the caller.

simulation_completed Logical. `TRUE` if the simulation ran for all requested days without early termination.

mortality_occurred Logical. `TRUE` if fish weight fell to or below 0.01 g during the simulation.

daily_output A `data.frame` with one row per simulated day containing temperature, consumption, respiration, egestion, excretion, net energy, weight, and energy density. Only present when `output_daily` = `TRUE`.

Examples

```
# Requires processed simulation data; see ?prepare_simulation_data
# sim <- run_fb4_simulation(
#   consumption_method = list(type = "p_value", value = 0.5),
#   processed_simulation_data = sim_data
# )
# sim$final_weight
```

`set_diet` *Set Diet Data for Bioenergetic Objects*

Description

Updates the diet data component of a Bioenergetic object with new diet composition and prey energy information.

Usage

```
set_diet(
  x,
  diet_proportions,
  prey_energies,
  indigestible_prey = NULL,
  normalize_diet = TRUE
)
```

Arguments

`x`	Bioenergetic object
`diet_proportions`	Data frame with daily diet proportions
`prey_energies`	Data frame with daily prey energy densities
`indigestible_prey`	Data frame with indigestible proportions (optional)
`normalize_diet`	Logical, whether to normalize diet proportions to sum to 1 (default TRUE)

Value

The `Bioenergetic` object `x` with its `diet_data` component updated (prey names, proportions, energies, and optionally indigestible fractions), and `fitted` reset to `FALSE`.

Examples

```
bio <- Bioenergetic(
  species_params = list(
    consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06)
  ),
  species_info = list(common_name = "Example fish")
)
diet  <- data.frame(Day = 1:365, prey1 = 0.6, prey2 = 0.4)
energ <- data.frame(Day = 1:365, prey1 = 4000, prey2 = 2500)
bio   <- set_diet(bio, diet, energ)
```

`set_environment` *Set Environmental Data for Bioenergetic Objects*

Description

Updates the environmental data component of a Bioenergetic object with new temperature information.

Usage

```
set_environment(x, temperature_data)
```

Arguments

`x`	Bioenergetic object
`temperature_data`	Data frame with Day and Temperature columns

Value

The `Bioenergetic` object `x` with its `environmental_data$temperature` component replaced by `temperature_data` (interpolated to fill missing days if needed), and `fitted` reset to `FALSE`.

Examples

```
bio <- Bioenergetic(
  species_params = list(
    consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06)
  ),
  species_info  = list(common_name = "Example fish")
)
```

```
temp <- data.frame(Day = 1:365, Temperature = rep(15, 365))
bio <- set_environment(bio, temp)
```

`set_parameter_value` *Set Parameter Value in Species Parameters*

Description

Sets a specific parameter value in species parameter lists, automatically finding the correct category.

Usage

```
set_parameter_value(params, param, value)
```

Arguments

`params`	Species parameters list
`param`	Parameter name to set
`value`	New parameter value

Value

The `params` list with `params[[category]][[param]]` replaced by `value`, where `category` is the first category in which `param` is found. Throws an error if `param` is not found in any category.

Examples

```
sp <- list(consumption = list(CA = 0.303, CB = -0.275))
updated <- set_parameter_value(sp, "CA", 0.350)
updated$consumption$CA
```

`set_simulation_settings` *Set Simulation Settings for Bioenergetic Objects*

Description

Updates the simulation configuration of a Bioenergetic object.

Usage

```
set_simulation_settings(x, initial_weight = NULL, duration = NULL)
```

Arguments

`x`	Bioenergetic object
`initial_weight`	Initial weight in grams
`duration`	Simulation duration in days (auto-detected if NULL)

Value

The `Bioenergetic` object `x` with `simulation_settings$initial_weight` and/or `simulation_settings$duration` updated, and `fitted` reset to `FALSE`. If `duration` is `NULL` and none was previously set, the duration is auto-detected from the maximum `Day` in environmental or diet data.

Examples

```
bio <- Bioenergetic(
  species_params = list(
    consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06)
  ),
  species_info = list(common_name = "Example fish")
)
bio <- set_simulation_settings(bio, initial_weight = 50, duration = 365)
bio$simulation_settings$initial_weight
```

`simulation-engine` *Simulation Engine for FB4 Model*

Description

Core daily simulation loop for the Fish Bioenergetics 4.0 model. The main entry point is `run_fb4_simulation`, which iterates over each simulated day calling low-level helpers for consumption (`calculate_daily_consumption`), metabolic losses (`calculate_daily_metabolism`), spawning costs (`calculate_daily_spawn_energy`), and weight change (`calculate_daily_weight_change`). Consumption can be specified as a proportion of maximum (`p_value`), a percentage of body weight (`ration_percent`), or absolute grams per day (`ration_grams`).

Value

No return value; this page documents the simulation engine functions module. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-binary-search` *Binary Search Strategy for FB4 Model*

Description

Implements the `"binary_search"` FB4 fitting strategy, which finds the proportion of maximum consumption (p-value) that minimises the difference between a simulated metric and a user-supplied target. Supported fitting targets are `"Weight"` (final weight in g) and `"Consumption"` (total consumption in g). The strategy is instantiated by `create_binary_search_strategy`; the search algorithm itself is in `binary_search_p_value`, coordinated by `fit_fb4_binary_search`.

Value

No return value; this page documents the binary search strategy functions. See individual function documentation for return values.

References

Hanson, P.C., Johnson, T.B., Schindler, D.E. and Kitchell, J.F. (1997). *Fish Bioenergetics 3.0*. University of Wisconsin Sea Grant Institute, Madison, WI.

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-bootstrap` *Bootstrap Estimation Strategy for FB4 Model*

Description

Implements the `"bootstrap"` FB4 fitting strategy, which estimates the proportion of maximum consumption (p-value) and its uncertainty by resampling observed final weights. Each bootstrap replicate finds the p_value that minimises the difference between the simulated final weight and the resampled mean weight via `optim_search_p_value`. Parallel execution is supported through the **future/furrr** ecosystem (`parallel = TRUE`).

Value

No return value; this page documents the bootstrap estimation strategy functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-commons` *Strategy Commons for FB4 Model*

Description

Internal helper functions shared across all FB4 fitting strategies. Provides a unified simulation executor (`execute_simulation_with_method`), a common parameter extractor (`extract_strategy_parameters`), metadata helpers (`add_strategy_metadata`), a final-simulation runner (`run_final_simulation`), input validation (`validate_common_strategy_inputs`), and a standardised error-result constructor (`create_error_result`).

Value

No return value; this page documents the common strategy utility functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-direct` *Direct Strategies for FB4 Model*

Description

Implements the `"direct"` FB4 execution strategy, which runs a single simulation with a user-supplied value instead of searching for an optimal parameter. Three input modes are supported: `"p_value"` (proportion of maximum consumption, 0–5), `"ration_percent"` (percentage of body weight per day, 0–100), and `"ration_grams"` (absolute daily ration in grams). The strategy is instantiated by `create_direct_strategy` and executed by `run_fb4_direct_method`.

Value

No return value; this page documents the direct p-value strategy functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-hierarchical` *Hierarchical Estimation Strategy for FB4 Model*

Description

Implements the `"hierarchical"` FB4 fitting strategy, which estimates population-level and individual-level p-values using a hierarchical mixed-effects model compiled with TMB. Individual p_values are treated as random effects (`log_p_individual`), while the population mean (`mu_p`) and standard deviation (`sigma_p`) are fixed-effect hyperparameters. Optional covariates can be supplied to explain variation in `mu_p` across individuals.

Value

No return value; this page documents the hierarchical estimation strategy functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H. and Bell, B.M. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5), 1–21. doi:10.18637/jss.v070.i05

`strategy-interface` *Strategy Interface and Factory for FB4 Model*

Description

Defines the common interface that all FB4 execution strategies must implement (`FB4Strategy`) and provides the factory function (`create_fb4_strategy`) that instantiates the correct strategy object based on the requested method. An execution plan is first assembled by `create_execution_plan`, validated for fit_to/strategy concordance by `validate_fit_to_strategy_concordance`, and then passed to the strategy's `execute()` method. Supported strategies are `"binary_search"`, `"optim"`, `"mle"`, `"bootstrap"`, `"hierarchical"`, and `"direct"` (with its ration variants).

Value

No return value; this page documents the strategy interface functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`strategy-mle` *Maximum Likelihood Estimation Strategy for FB4 Model*

Description

Implements the `"mle"` FB4 fitting strategy, which estimates the proportion of maximum consumption (p-value) and its uncertainty by maximising the log-normal likelihood of observed final weights. Both an R backend (`fit_fb4_mle`) and a faster TMB/C++ backend (`execute_mle_tmb`) are supported. Confidence intervals are derived from the Hessian (delta method) and optionally from a likelihood profile (`compute_likelihood_profile`).

Value

No return value; this page documents the maximum likelihood estimation strategy functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H. and Bell, B.M. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5), 1–21. doi:10.18637/jss.v070.i05

`strategy-optim` *Optimisation Strategy for FB4 Model*

Description

Implements the `"optim"` FB4 fitting strategy, which uses R's `optim` function to find the proportion of maximum consumption (p-value) that minimises the difference between a simulated metric and a user-supplied target. Supported fitting targets are `"Weight"` and `"Consumption"`. The strategy is instantiated by `create_optim_strategy`; the optimisation algorithm is in `optim_search_p_value`, coordinated by `fit_fb4_optim`.

Value

No return value; this page documents the optimisation-based strategy functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`summary.Bioenergetic` *Summary Method for Bioenergetic Objects*

Description

Prints a detailed multi-section summary of a `Bioenergetic` object, covering species identity, parameter categories, environmental data statistics, diet composition, simulation settings, and overall readiness status. Complements `print.Bioenergetic`, which shows the compact single-page view.

Usage

```
## S3 method for class 'Bioenergetic'
summary(object, ...)
```

Arguments

`object`	Bioenergetic object
`...`	Additional arguments (not used)

Value

Invisibly returns the input object

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
summary(bio)
```

`summary.fb4_result` *Summary Method for fb4_result Objects*

Description

Prints the compact `print.fb4_result` output followed by a detailed section with execution metadata (backend, version, timing) and method-specific diagnostics: optimisation tolerances for traditional methods, statistical details and profile-likelihood availability for MLE, success rate and percentiles for bootstrap, and individual/population parameter distributions for hierarchical fits. Daily output column availability is reported at the end.

Usage

```
## S3 method for class 'fb4_result'
summary(object, ...)
```

Arguments

`object`	fb4_result object
`...`	Additional arguments (not used)

Value

Invisibly returns the input object

Examples

```
data(fish4_parameters)
sp   <- fish4_parameters[["Oncorhynchus tshawytscha"]]$life_stages$adult
info <- fish4_parameters[["Oncorhynchus tshawytscha"]]$species_info
bio  <- Bioenergetic(
  species_params     = sp,
  species_info       = info,
  environmental_data = list(
    temperature = data.frame(Day = 1:30, Temperature = rep(12, 30))
  ),
  diet_data = list(
    proportions = data.frame(Day = 1:30, Prey1 = 1.0),
    energies    = data.frame(Day = 1:30, Prey1 = 5000),
    prey_names  = "Prey1"
  ),
  simulation_settings = list(initial_weight = 100, duration = 30)
)
bio$species_params$predator$ED_ini <- 5000
bio$species_params$predator$ED_end <- 5500
result <- run_fb4(bio, strategy = "direct", p_value = 0.5, verbose = FALSE)
summary(result)
```

`uncertainty-prediction`

FB4 Uncertainty Propagation Functions

Description

Functions for propagating *p*-value uncertainty to consumption predictions using two approaches: the delta method (`predict_consumption_delta`), which applies a first-order Taylor series approximation and is efficient when the consumption-p relationship is approximately linear; and a parametric bootstrap (`predict_consumption_bootstrap`), which samples from the estimated p-value distribution, runs a full FB4 simulation for each sample, and derives the full consumption uncertainty distribution without linearity assumptions.

Value

No return value; this page documents the uncertainty and prediction functions. See individual function documentation for return values.

References

Deslauriers, D., Chipps, S.R., Breck, J.E., Rice, J.A. and Madenjian, C.P. (2017). Fish Bioenergetics 4.0: An R-based modeling application. *Fisheries*, 42(11), 586–596. doi:10.1080/03632415.2017.1377558

`update_body_composition`

Update body composition during simulation (Mid-level)

Description

Updates body composition as fish grows or changes condition Used during simulation loops - assumes pre-validated inputs

Usage

```
update_body_composition(
  old_weight,
  new_weight,
  old_composition = NULL,
  processed_composition_params
)
```

Arguments

`old_weight`	Previous weight (g)
`new_weight`	New weight (g)
`old_composition`	Previous composition (optional)
`processed_composition_params`	List with processed composition parameters

Value

A named list with the same 13 elements as `calculate_body_composition`, describing the body composition at `new_weight`. If `old_composition` is supplied, an additional element `changes` is appended — a named list with five numeric scalars: `weight_change`, `water_change`, `protein_change`, `fat_change`, and `energy_density_change` (all in the same units as the corresponding composition elements).

Examples

```
params <- list(water_fraction = 0.72, fat_energy = 36450,
               protein_energy = 17990, max_fat_fraction = 0.30)
old <- calculate_body_composition(weight = 100,
                                  processed_composition_params = params)
update_body_composition(old_weight = 100, new_weight = 110,
                        old_composition = old,
                        processed_composition_params = params)
```

`utils` *Utility Functions for fb4package*

Description

Cross-cutting utilities used throughout the package: the null-coalescing operator, safe mathematical operations, and value clamping. These functions have no internal dependencies and are available to all layers.

Value

No return value; this page documents cross-cutting utility functions used throughout the package. See individual function documentation for return values.

`validate_basic_params` *Validate Basic Model Parameters*

Description

Validates fundamental model parameters for biological feasibility and computational practicality.

Usage

```
validate_basic_params(initial_weight, duration)
```

Arguments

`initial_weight`	Initial weight in grams
`duration`	Duration in days

Details

Checks that:

- Initial weight is positive and numeric
- Duration is positive and numeric
- Duration is not excessively long (performance warning)

Value

Invisibly returns `TRUE` if validation passes; throws an error otherwise.

Examples

```
isTRUE(validate_basic_params(10.5, 365))
try(validate_basic_params(-5, 100))
```

`validate_bioenergetic_for_simulation` *Comprehensive validation for Bioenergetic objects*

Description

Validates a Bioenergetic object before simulation, checking all required components with comprehensive error accumulation.

Usage

```
validate_bioenergetic_for_simulation(bio_obj)
```

Arguments

`bio_obj`	Bioenergetic object

Value

A named list with four elements: `valid` (logical), `errors` (character vector), `warnings` (character vector), and `ready_to_run` (logical, `TRUE` only when `valid` is `TRUE`). Errors are accumulated across all sub-checks (structure, species equations, temperature data, diet data, initial weight) rather than stopping at the first failure.

Examples

```
# Requires a fully-configured Bioenergetic object; see ?Bioenergetic
# result <- validate_bioenergetic_for_simulation(bio)
# result$valid
```

`validate_body_composition`

Validate body composition

Description

Validate body composition

Usage

```
validate_body_composition(composition)
```

Arguments

`composition`	Body composition list

Value

A named list with three elements: `valid` (logical), `errors` (character vector), and `warnings` (character vector). `valid` is `FALSE` if required composition fields are missing or if fractions do not sum to approximately 1. `warnings` are issued when individual fractions fall outside typical biological ranges for fish.

Examples

```
comp <- calculate_body_composition(
  weight = 100,
  processed_composition_params = list(water_fraction = 0.72,
    fat_energy = 36450, protein_energy = 17990, max_fat_fraction = 0.30)
)
validate_body_composition(comp)$valid
```

`validate_contaminant_params` *Validate contaminant parameters*

Description

Validate contaminant parameters

Usage

```
validate_contaminant_params(contaminant_params)
```

Arguments

`contaminant_params`	List with parameters

Value

A named list with three elements: `valid` (logical), `errors` (character vector), and `warnings` (character vector). `valid` is `FALSE` if `CONTEQ` is not 1–3 or if prey concentrations or efficiency values fail range checks.

Examples

```
validate_contaminant_params(list(
  CONTEQ = 1,
  prey_concentrations = c(0.05, 0.08),
  transfer_efficiency = c(0.8, 0.8)
))$valid
```

`validate_fb4_inputs` *Validate inputs for FB4 simulation*

Description

Validates all inputs for FB4 simulation, including the Bioenergetic object and strategy-specific parameters.

Usage

```
validate_fb4_inputs(
  bio_obj,
  strategy,
  fit_to = NULL,
  fit_value = NULL,
  first_day = 1,
  last_day = NULL,
  observed_weights = NULL,
  covariates = NULL
)
```

Arguments

`bio_obj`	Bioenergetic object
`strategy`	Strategy to use: "binary_search", "optim", "bootstrap", "mle", "hierarchical"
`fit_to`	Fitting target (for traditional strategies)
`fit_value`	Fitting value (for traditional strategies)
`first_day`	First simulation day
`last_day`	Last simulation day
`observed_weights`	Vector of observed weights (for statistical strategies)
`covariates`	Covariates (for hierarchical strategy)

Value

Invisibly returns `TRUE` if all inputs are valid. Throws an error with a descriptive message at the first validation failure: invalid day range, unrecognised strategy, missing `fit_to`/`fit_value` for traditional strategies, or missing `observed_weights` for statistical strategies.

Examples

```
# Requires a fully-configured Bioenergetic object; see ?Bioenergetic
# validate_fb4_inputs(bio, strategy = "direct",
#                     fit_to = "Weight", fit_value = 200,
#                     first_day = 1, last_day = 365)
```

`validate_fb4_system` *Validate complete FB4 system ready for simulation*

Description

Comprehensive validation that combines all validation layers. This is the ultimate validation function for production use.

Usage

```
validate_fb4_system(
  bio_obj,
  strategy,
  first_day = 1,
  last_day = NULL,
  validation_level = "standard",
  ...
)
```

Arguments

`bio_obj`	Bioenergetic object
`strategy`	Strategy to use: "binary_search", "optim", "bootstrap", "mle", "hierarchical"
`first_day`	First simulation day
`last_day`	Last simulation day
`validation_level`	Validation strictness ("basic", "standard", "comprehensive")
`...`	Additional arguments for strategy-specific validation

Value

A named list with seven elements: `system_valid` (logical), `validation_level` (character), `errors` (character vector), `warnings` (character vector), `info` (character vector), `component_results` (named list with results from each validation layer), and `timestamp` (POSIXct). `system_valid` is `TRUE` only when all active validation layers pass.

Examples

```
# Requires a fully-configured Bioenergetic object; see ?Bioenergetic
# result <- validate_fb4_system(bio, strategy = "direct")
# result$system_valid
```

`validate_fraction` *Validate fraction values (0-1 range)*

Description

Specialized validator for fraction/proportion values.

Usage

```
validate_fraction(
  value,
  param_name,
  strategy = "strict",
  allow_zero = TRUE,
  allow_one = TRUE
)
```

Arguments

`value`	Value(s) to validate
`param_name`	Parameter name
`strategy`	Handling strategy
`allow_zero`	Whether zero is allowed
`allow_one`	Whether one is allowed

Value

An object of class `fb4_validation` (see `validation_result`). `valid` is `TRUE` when all values lie within $[0, 1]$ (or the bounds set by `allow_zero` and `allow_one`). Out-of-range values are recorded in `errors` (strategy `"strict"`) or `warnings` (strategy `"warn"`).

Examples

```
validate_fraction(0.5, "diet_proportion")
validate_fraction(c(0.3, 0.7), "fractions")
```

`validate_individual_data`
Validate individual data for hierarchical models

Description

Validates individual fish data for hierarchical mark-recapture models.

Usage

```
validate_individual_data(
  individual_data,
  require_positive_growth = TRUE,
  check_outliers = TRUE
)
```

Arguments

`individual_data`	Data frame with individual observations
`require_positive_growth`	Whether growth must be positive
`check_outliers`	Whether to check for outliers

Value

Invisibly returns `TRUE` if all checks pass. Throws an error if `individual_data` is not a data frame, if required columns (`individual_id`, `initial_weight`, `final_weight`) are missing, if weights are non-positive, or if growth is negative when `require_positive_growth` = `TRUE`. Issues warnings for outlier observations when `check_outliers` = `TRUE`.

Examples

```
ind <- data.frame(individual_id = c("A", "B"),
                  initial_weight = c(50, 80),
                  final_weight   = c(120, 180))
isTRUE(validate_individual_data(ind))
```

`validate_nutrient_concentrations`

Validate nutrient concentrations

Description

Validate nutrient concentrations

Usage

```
validate_nutrient_concentrations(
  nutrient_concentrations,
  organism_type = "fish"
)
```

Arguments

`nutrient_concentrations`	List with N and P concentrations
`organism_type`	Organism type for validation

Value

A named list with three elements: `valid` (logical), `errors` (character vector), and `warnings` (character vector). `warnings` are issued when N or P concentrations fall outside the typical range for the specified `organism_type` or when the N:P mass ratio is outside 2–20.

Examples

```
validate_nutrient_concentrations(list(
  nitrogen   = 0.030,
  phosphorus = 0.004
))$valid
```

`validate_positive` *Validate positive values*

Description

Specialized validator for positive numeric values.

Usage

```
validate_positive(value, param_name, strategy = "strict", min_val = 0.001)
```

Arguments

`value`	Value(s) to validate
`param_name`	Parameter name
`strategy`	Handling strategy
`min_val`	Minimum positive value (default 0.001)

Value

An object of class `fb4_validation` (see `validation_result`). `valid` is `TRUE` when all values are $\geq$ `min_val`. Violations are recorded in `errors` (`strategy = "strict"`) or `warnings` (`strategy = "warn"`).

Examples

```
validate_positive(5, "weight")
validate_positive(0, "weight")$valid
```

`validate_predator_energy_params` *Validate predator energy density parameters*

Description

Validate predator energy density parameters

Usage

```
validate_predator_energy_params(predator_params, weight_range = c(1, 1000))
```

Arguments

`predator_params`	List with parameters
`weight_range`	Weight range for testing

Value

A named list with three elements: `valid` (logical), `errors` (character vector), and `warnings` (character vector). `valid` is `FALSE` if `PREDEDEQ` is not 1–3, if parameter calculations fail, or if required parameters are missing. `warnings` may flag energy densities outside the typical 1000–15000 J/g range.

Examples

```
validate_predator_energy_params(
  list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1)
)
```

`validate_species_equations` *Main function to validate all species equations*

Description

Main function to validate all species equations

Usage

```
validate_species_equations(species_params)
```

Arguments

`species_params`	List with all species parameters

Value

A named list with four elements: `valid` (logical), `errors` (character vector), `warnings` (character vector), and `category_results` (named list with one validation result per bioenergetic category checked).

Examples

```
sp <- list(
  consumption = list(CEQ = 1, CA = 0.303, CB = -0.275, CQ = 0.06),
  respiration = list(REQ = 2, RA = 0.0033, RB = -0.227,
                     RQ = 0.025, RTM = 30, RTO = 18),
  egestion    = list(EGEQ = 1, FA = 0.16),
  excretion   = list(EXEQ = 1, UA = 0.10),
  predator    = list(PREDEDEQ = 3, Alpha1 = 4800, Beta1 = 0.1)
)
validate_species_equations(sp)$valid
```

`validate_temperature` *Validate temperature values*

Description

Specialized validator for temperature values with realistic ranges.

Usage

```
validate_temperature(
  value,
  param_name,
  strategy = "warn",
  min_temp = -5,
  max_temp = 45
)
```

Arguments

`value`	Temperature value(s) in Celsius
`param_name`	Parameter name
`strategy`	Handling strategy
`min_temp`	Minimum realistic temperature (default -5°C)
`max_temp`	Maximum realistic temperature (default 45°C)

Value

An object of class `fb4_validation` (see `validation_result`). `valid` is `TRUE` when all values are finite and lie within `[min_temp, max_temp]`. Values outside the range are recorded in `warnings` by default (`strategy = "warn"`).

Examples

```
validate_temperature(15, "water_temp")
validate_temperature(c(5, 12, 18), "temperatures")
```

`validate_time_series_data`

Validate Time Series Data Structure (Basic Level)

Description

Basic validation of time series data structure and content for use in FB4 simulations.

Usage

```
validate_time_series_data(
  data,
  data_name,
  required_cols = NULL,
  min_cols = NULL
)
```

Arguments

`data`	Data to validate
`data_name`	Name of the dataset (for error messages)
`required_cols`	Required column names
`min_cols`	Minimum number of columns

Details

Performs comprehensive validation including:

- Structure validation (data.frame, non-empty)
- Required column presence
- Day column validation (numeric, finite, ascending)
- Duplicate detection

Value

Invisibly returns `TRUE` if validation passes; throws an error otherwise.

Examples

```
temp_data <- data.frame(Day = 1:10, Temperature = 15:24)
isTRUE(validate_time_series_data(temp_data, "temperature", c("Day", "Temperature")))
```

`validation_result` *Create standardized validation result*

Description

Constructor for standardized validation result objects used throughout the FB4 validation system.

Usage

```
validation_result(
  valid = TRUE,
  errors = character(),
  warnings = character(),
  info = character(),
  level = "core",
  category = NULL,
  checked_items = list()
)
```

Arguments

`valid`	Logical indicating if validation passed
`errors`	Character vector of error messages
`warnings`	Character vector of warning messages
`info`	Character vector of info messages
`level`	Validation level ("core", "structure", "parameter", etc.)
`category`	Optional category being validated
`checked_items`	List of items that were checked

Value

An object of class `fb4_validation`: a named list with eight elements: `valid` (logical), `errors` (character vector of error messages), `warnings` (character vector of warning messages), `info` (character vector of informational messages), `level` (character, validation tier), `category` (character or `NULL`), `checked_items` (list of items examined), and `timestamp` (POSIXct).

Examples

```
validation_result(valid = TRUE)
validation_result(valid = FALSE, errors = "weight must be positive",
                  level = "parameter")
```

Index