

Package: kelpbio (via r-universe)

October 20, 2026

Title Bayesian Kelp Biomass Estimation

Version 0.0.0.9000

Description Fit a Bayesian hierarchical Stan model via 'rstan' to estimate the allometric relationship between kelp weight and size from in-situ observations.

License MIT + file LICENSE

URL <https://github.com/HakaiInstitute/kelpbio>,
<https://hakaiinstitute.github.io/kelpbio/>

BugReports <https://github.com/HakaiInstitute/kelpbio/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests bayesplot, knitr, loo, rmarkdown, testthat (>= 3.0.0),
vdiff, withr

VignetteBuilder knitr

Config/testthat/edition 3

Biarch true

Depends R (>= 4.2.0)

Imports callr, chk, cli, dplyr, extras, generics, ggplot2, methods,
parallel, posterior, purrr, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.32.0), rstantools (>= 2.4.0),
tibble, universals

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),
RcppParallel (>= 5.0.1), rstan (>= 2.32.0), StanHeaders (>= 2.32.0)

SystemRequirements GNU make

LazyData true

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make

Repository <https://poissonconsulting.r-universe.dev>

Date/Publication 2026-07-25 02:02:20 UTC
RemoteUrl https://github.com/HakaiInstitute/kelpbio
RemoteRef HEAD
RemoteSha ca11bef838a27cbde1a080566b3e8afbf3bb44ee

Contents

augment.kb_fit	2
autoplot.kb_predictions	3
coef.kb_fit	4
converged.kb_fit	5
data_weight_sim_nereo	6
fit_weight_sim_nereo	6
fitted.kb_fit_weight	7
glance.kb_fit	8
kb_check_data_weight_nereo	9
kb_fit_progress	10
kb_fit_weight_nereo	10
kb_plot_predictions	12
kb_predict_weight	14
kb_predict_weight_by	15
kb_prior_exponential	17
kb_prior_normal	18
kb_priors_weight_nereo	18
kb_stancode	19
log_lik.kb_fit	20
posterior_epred.kb_fit_weight	20
posterior_linpred.kb_fit_weight	22
posterior_predict.kb_fit_weight	23
predict.kb_fit_weight	24
prior_summary.kb_fit	26
residuals.kb_fit_weight	27
samples	28
summary.kb_fit	28
tidy.kb_fit_weight	30
Index	32

augment.kb_fit	<i>Augment Model Data</i>
----------------	---------------------------

Description

Append the `fitted()` and deviance `residuals()` values to the input data. Values are the point estimate (median) of the posterior distributions.

Usage

```
## S3 method for class 'kb_fit'
augment(x, ...)
```

Arguments

x	A kb_fit object.
...	Unused.

Value

The input data with added columns fitted (response-scale fitted value) and residual (deviance residual).

See Also

[fitted\(\)](#) and [residuals\(\)](#).

Other generics: [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
augment(fit_weight_sim_nereo)
```

```
autoplot.kb_predictions
```

Autoplot Method for Predictions

Description

The conventional `ggplot2::autoplot` entry point for a `kb_predictions` object: a thin wrapper on [kb_plot_predictions\(\)](#). Dispatches on the prediction data frame, never on a fit.

Usage

```
## S3 method for class 'kb_predictions'
autoplot(object, ...)
```

Arguments

object	A kb_predictions object.
...	Passed to kb_plot_predictions() (e.g. x, observed).

Value

A ggplot object.

See Also

Other prediction: [kb_plot_predictions\(\)](#), [kb_predict_weight\(\)](#), [kb_predict_weight_by\(\)](#)

Examples

```
p <- kb_predict_weight_by(fit_weight_sim_nereo, by = "site")
ggplot2::autoplot(p)
```

coef.kb_fit	<i>Coefficients of a Model Fit</i>
-------------	------------------------------------

Description

A wrapper on [tidy\(\)](#) returning the model-term posterior summaries.

Usage

```
## S3 method for class 'kb_fit'
coef(object, ...)
```

Arguments

```
object      A kb_fit object.
...         Passed to tidy\(\).
```

Value

A tibble with one row per term and columns term, estimate, lower, and upper.

See Also

Other generics: [augment.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
coef(fit_weight_sim_nereo)
```

converged.kb_fit	<i>Convergence of a Model Fit</i>
------------------	-----------------------------------

Description

Whether a model fit has converged, from the Rhat and bulk effective sample rate of its parameters.

Usage

```
## S3 method for class 'kb_fit'
converged(x, ..., rhat = 1.05, esr = 0.1)
```

Arguments

x	A kb_fit object.
...	Unused.
rhat	A number giving the maximum acceptable Rhat.
esr	A number giving the minimum acceptable effective sample rate (effective sample size divided by the number of draws).

Details

esr is the effective sample rate: the bulk effective sample size divided by the number of draws. Convergence is assessed on the bulk effective sample size only, not the tail. The defaults are rhat = 1.05 and esr = 0.1.

Value

A flag: TRUE if all Rhat are below rhat and all bulk effective sample rates above esr.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
converged(fit_weight_sim_nereo)
```

`data_weight_sim_nereo` *Simulated Nereocystis Weight Dataset*

Description

A small simulated dataset of sub-bulb diameter and wet weight (columns diameter, weight, site, year), for fast tests and runnable examples. It is simulated from the weight-model structure (site intercept, site slope, and site:year random effects over a wide diameter range), not real survey data, and is not intended for inference. Built by `data-raw/data_weight_sim_nereo.R`.

Usage

```
data_weight_sim_nereo
```

Format

A data frame with columns:

diameter Sub-bulb diameter (mm), a positive number.

weight Wet weight (kg), a positive number.

site Survey site, a factor (10 levels).

year Survey year, a factor (4 levels).

See Also

[fit_weight_sim_nereo](#) for a fit to this dataset.

Other data: [fit_weight_sim_nereo](#), [kb_check_data_weight_nereo\(\)](#)

`fit_weight_sim_nereo` *Example Weight Model Fit*

Description

A slim pre-fit weight model, the result of [kb_fit_weight_nereo\(\)](#) on [data_weight_sim_nereo](#) with a reduced number of chains and draws. Provided so the examples and tests for the fit accessors, S3 methods, and prediction functions run quickly. It is fit to simulated data and is not intended for inference. Built by `data-raw/fit_weight_sim_nereo.R`.

Usage

```
fit_weight_sim_nereo
```

Format

An object of class `c("kb_fit_weight", "kb_fit")`.

See Also

[kb_fit_weight_nereo\(\)](#) and [data_weight_sim_nereo](#).

Other data: [data_weight_sim_nereo](#), [kb_check_data_weight_nereo\(\)](#)

fitted.kb_fit_weight *Fitted Weights*

Description

Posterior point estimates (median) of the expected weight at each observed row, on the response scale, matching [augment\(\)](#)'s fitted column. For the full posterior, use [posterior_epred\(\)](#).

Usage

```
## S3 method for class 'kb_fit_weight'
fitted(object, ...)
```

Arguments

object	A kb_fit_weight object.
...	Unused.

Value

A numeric vector of fitted weights, length `nobs(object)`.

See Also

[residuals\(\)](#) for deviance residuals, [augment\(\)](#), and [posterior_epred\(\)](#) for the full posterior.

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
fitted(fit_weight_sim_nereo)
```

glance.kb_fit	<i>Model Fit Diagnostic Summary</i>
---------------	-------------------------------------

Description

One-row summary of a model fit with convergence diagnostics.

Usage

```
## S3 method for class 'kb_fit'
glance(x, ..., rhat = 1.05, esr = 0.1)
```

Arguments

x	A kb_fit object.
...	Unused.
rhat	A number giving the maximum acceptable Rhat.
esr	A number giving the minimum acceptable effective sample rate (effective sample size divided by the number of draws).

Value

A one-row tibble with n, K, nchains, niters, nthin, ess (the minimum bulk effective sample size), rhat (the maximum Rhat), and converged.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
glance(fit_weight_sim_nereo)
```

`kb_check_data_weight_nereo`*Validate Nereocystis Weight Model Data*

Description

Check that data contains the columns required to fit the *Nereocystis luetkeana* weight model, with appropriate types and values.

Usage

```
kb_check_data_weight_nereo(  
  data,  
  x_name = chk::deparse_backtick_chk(substitute(data))  
)
```

Arguments

<code>data</code>	A data frame of weight observations (see kb_check_data_weight_nereo() for the required columns).
<code>x_name</code>	A string naming data in error messages.

Details

Required columns: numeric diameter (> 0), numeric weight (> 0), and factor or character site and year, with no missing values.

Diameter and weight may be in any units, provided prediction data use the same units as the fitted data.

Value

`data`, invisibly.

See Also

Other data: [data_weight_sim_nereo](#), [fit_weight_sim_nereo](#)

Examples

```
data <- data.frame(  
  diameter = c(20, 35), weight = c(0.5, 2.1),  
  site = factor(c("a", "b")), year = factor(c("2020", "2021"))  
)  
kb_check_data_weight_nereo(data)
```

kb_fit_progress	<i>Completed Fraction of a Running Fit</i>
-----------------	--

Description

The proportion of sampling completed by a fit writing to progress_dir, as a number from 0 to 1.

Usage

```
kb_fit_progress(progress_dir)
```

Arguments

progress_dir A string giving the directory passed as progress_dir to a kb_fit_* function.

Details

Pass the same directory given as progress_dir to a kb_fit_* function. This reads fit progress from another R process: for example, a Shiny app can run the fit in a background task and poll kb_fit_progress() from the main session to drive a progress indicator. It returns 0 before sampling has produced any output (or when the directory holds no fit artifact yet) and 1 once every chain has finished.

Value

A number between 0 and 1.

Examples

```
progress_dir <- tempfile()
dir.create(progress_dir)
# Before a fit has written anything, progress is 0.
kb_fit_progress(progress_dir)
```

kb_fit_weight_nereo	<i>Fit a Nereocystis Weight Model</i>
---------------------	---------------------------------------

Description

Fit an allometric weight model for *Nereocystis luetkeana* via Stan.

Usage

```

kb_fit_weight_nereo(
  data,
  priors = NULL,
  ...,
  prior_only = FALSE,
  chains = 4L,
  niters = 1000L,
  nthin = 1L,
  cores = NULL,
  seed = NULL,
  progress = c("bar", "verbose", "none"),
  progress_dir = NULL
)

```

Arguments

data	A data frame of weight observations (see kb_check_data_weight_nereo() for the required columns).
priors	A named list of prior objects (see kb_priors_weight_nereo()), or NULL to use the defaults. Supplied entries override the corresponding defaults; unspecified entries keep their defaults.
...	Additional arguments passed to rstan::sampling() , including a control list (merged over the <code>adapt_delta = 0.95</code> default); see the control argument of rstan::stan() for the available entries.
prior_only	A flag specifying whether to sample from the priors only (the likelihood is switched off), for prior predictive checks.
chains	A whole number of MCMC chains.
niters	A whole number of saved post-warmup draws per chain (warmup defaults to match).
nthin	A whole number giving the thinning interval.
cores	A whole number of cores for parallel chains, or NULL to use <code>getOption("mc.cores")</code> (falling back to chains), capped at the available cores.
seed	A whole number passed to rstan::sampling() to make the fit reproducible, or NULL (the default). When NULL, a seed is drawn from R's RNG, so a preceding <code>set.seed()</code> also makes the fit reproducible; an explicit seed takes precedence over the RNG state.
progress	A string, one of "bar" (the default, a console progress bar), "verbose" (rstan's per-iteration output and diagnostic warnings), or "none" (silent). Controls fit-time console output only.
progress_dir	A string giving an existing directory in which to write a pollable progress artifact (read by kb_fit_progress()), or NULL (the default) to write none.

Details

The response is log wet weight, modelled with a Student-t likelihood (4 degrees of freedom, for robustness to outliers). Expected log weight is a quadratic (allometric) function of log sub-bulb diameter, centered at its geometric mean so the intercept is the expected log weight at a typical diameter. The intercept and the allometric slope vary by site, and the intercept also varies by site:year.

niters is the number of saved post-warmup draws per chain; warmup defaults to match niters and the post-warmup phase is thinned by nthin. The returned object stores the extracted posterior draws (including the log_lik and yrep generated quantities), diagnostics, data, and metadata.

progress controls fit-time console output. The default "bar" shows a progress bar; "verbose" streams rstan's per-iteration output and its post-sampling diagnostic warnings; "none" is silent. progress changes only console output, never the fit; inspect convergence with [converged\(\)](#) / [glance\(\)](#) / [summary\(\)](#) in every mode.

Supply progress_dir (an existing directory) to have the fit write a pollable progress artifact there, which [kb_fit_progress\(\)](#) reads to report the completed fraction from another R process (for example, to drive a progress indicator while the fit runs in the background).

Chains run in parallel by default (cores = NULL uses `getOption("mc.cores")`), falling back to chains, capped at the available cores). Set `options(mc.cores = 1)` (or pass `cores = 1`) on shared servers, in containers, or inside another parallel context.

The sampler runs with a conservative `adapt_delta = 0.95` by default, which reduces divergences at the cost of slightly longer runtime. Override it, or set any other sampler control, by passing a control list through ..., e.g. `kb_fit_weight_nereo(data, control = list(adapt_delta = 0.99))`; only the entries supplied are changed. See the control argument of [rstan::stan\(\)](#) for the full set of tunable entries (e.g. `adapt_delta`, `max_treedepth`).

The site:year effect is set from the data: it is dropped when the data span a single year (then confounded with the site effect) and included otherwise. When no site spans more than one year it is kept with a warning, since `sSite` and `sSiteYear` are not then separately identified.

Value

An object of class `c("kb_fit_weight", "kb_fit")`.

Examples

```
if (interactive()) {
  fit <- kb_fit_weight_nereo(data_weight_sim_nereo)
  tidy(fit)
}
# A pre-fit example model is included with the package:
tidy(fit_weight_sim_nereo)
```

kb_plot_predictions	<i>Plot Model Predictions</i>
---------------------	-------------------------------

Description

Render a ggplot from a `kb_predictions` object (the output of a `kb_predict_*`() function).

Usage

```
kb_plot_predictions(
  predictions,
  ...,
  x = NULL,
  observed = NULL,
  max_facets = 12L
)
```

Arguments

<code>predictions</code>	A <code>kb_predictions</code> object.
<code>...</code>	Unused.
<code>x</code>	A string naming the x-axis column, or <code>NULL</code> to infer it from the metadata.
<code>observed</code>	A data frame of raw observations to overlay as points, or <code>NULL</code> for none.
<code>max_facets</code>	A whole number capping the facet panels drawn; if the grouping has more groups, the first <code>max_facets</code> are shown with a warning. Use <code>Inf</code> to disable.

Details

Called with just the predictions, the plot configures itself: the x-axis, faceting, and geometry are inferred from the prediction's metadata. The result is a standard `ggplot` that can be refined with `+` (scales, labels, themes, `coord_flip()`).

The geometry is chosen automatically, not set by an argument: a line with a compatibility-interval ribbon for a generated curve (`kb_predict_weight_by()` over a varying predictor), and `geom_pointrange` otherwise. Override the inferred x-axis with `x`, and overlay the raw data with `observed`.

Value

A `ggplot` object.

See Also

Other prediction: [autoplot.kb_predictions\(\)](#), [kb_predict_weight\(\)](#), [kb_predict_weight_by\(\)](#)

Examples

```
# Allometric curve by site (ribbon):
kb_predict_weight_by(fit_weight_sim_nereo, by = "site") |>
  kb_plot_predictions()

# Weight at a reference diameter by site (pointrange, sites on the y-axis):
kb_predict_weight_by(
  fit_weight_sim_nereo,
  by = "site", diameter = 30, new_levels = "average"
) |>
  kb_plot_predictions() +
  ggplot2::coord_flip()
```

kb_predict_weight	<i>Predict Weight for New Data</i>
-------------------	------------------------------------

Description

Predict weight for the supplied rows, or for the observed data when `new_data = NULL`. For an allometric curve over a diameter sequence, use `kb_predict_weight_by()` instead.

Usage

```
kb_predict_weight(
  fit,
  new_data = NULL,
  ...,
  new_levels = c("sample", "average"),
  representative_site = NULL,
  conf_level = 0.95,
  estimate = stats::median,
  sig_fig = 3
)
```

Arguments

<code>fit</code>	A <code>kb_fit_weight</code> object.
<code>new_data</code>	A data frame with a diameter column (and optional site / year columns), or NULL to predict at the observed data.
<code>...</code>	These dots are for future extensions and must be empty.
<code>new_levels</code>	A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with <code>set.seed()</code> for a reproducible interval.
<code>representative_site</code>	A character vector of site levels present in the fit, or NULL (the default). When supplied, a new or absent site takes its site main effects (intercept and slope) from the named reference site (the per-draw average when several are named), instead of the <code>new_levels</code> treatment; the <code>site:year</code> interaction still follows <code>new_levels</code> .
<code>conf_level</code>	A number between 0 and 1 giving the compatibility-interval level.
<code>estimate</code>	A function that reduces a numeric vector of posterior draws to a scalar point estimate (e.g. median or mean).
<code>sig_fig</code>	A whole number of significant figures for summary output.

Details

Conditioning is resolved per row: a site/year value the model has seen is conditioned on its estimated random effects; a new site or year, or an absent grouping column, is handled by new_levels.

With the default new_levels = "sample", an absent or new group draws a random effect from its estimated distribution, so the interval includes between-group variation; "average" instead holds those random effects at zero. "sample" draws fresh values on each call; set a seed with `set.seed()` for a reproducible interval. "sample" is the default because it produces honest uncertainty for a new, unobserved site/year.

For a new site, representative_site offers a third approach: instead of new_levels ("sample" or "average") it borrows the site intercept and slope of one or more named reference sites (the per-draw average across several). The site:year interaction still follows new_levels.

Value

A kb_predictions object: the input rows with added estimate, lower, and upper columns.

See Also

`kb_predict_weight_by()` to generate new_data by grouping factors and diameter sequence, and `augment()` for fitted/residual values at the observed data.

Other prediction: `autoplot.kb_predictions()`, `kb_plot_predictions()`, `kb_predict_weight_by()`

Examples

```
new_data <- data.frame(diameter = c(20, 40, 60))
kb_predict_weight(fit_weight_sim_nereo, new_data, new_levels = "average")

# Predict a new site as if it behaves like a known reference site:
new_site <- data.frame(diameter = c(20, 40, 60), site = "new_site")
kb_predict_weight(
  fit_weight_sim_nereo, new_site,
  representative_site = fit_weight_sim_nereo$meta$site_levels[1]
)
```

kb_predict_weight_by *Predict Weight Over a Diameter Sequence by Grouping Factor*

Description

Summarise the fitted relationship between weight and diameter over a generated prediction grid: a sequence of diameter values crossed with the grouping factors named in by (one curve per group).

Usage

```
kb_predict_weight_by(
  fit,
  by = NULL,
  diameter = NULL,
  ...,
  new_levels = c("average", "sample"),
  conf_level = 0.95,
  estimate = stats::median,
  sig_fig = 3
)
```

Arguments

<code>fit</code>	A <code>kb_fit_weight</code> object.
<code>by</code>	A character vector of grouping factors, each drawn as a separate curve, or <code>NULL</code> for a single population-level curve. Each named factor is expanded over its observed levels and conditioned on its estimated random effects.
<code>diameter</code>	A numeric vector of sub-bulb diameters to predict over (in the same units as the fitted data), or <code>NULL</code> for an automatic sequence spanning the observed range.
<code>...</code>	These dots are for future extensions and must be empty.
<code>new_levels</code>	A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with <code>set.seed()</code> for a reproducible interval.
<code>conf_level</code>	A number between 0 and 1 giving the compatibility-interval level.
<code>estimate</code>	A function that reduces a numeric vector of posterior draws to a scalar point estimate (e.g. median or mean).
<code>sig_fig</code>	A whole number of significant figures for summary output.

Details

`by` selects the grouping factors that each get their own curve, conditioned on their estimated random effects. `new_levels` controls the factors not named in `by`. The default "average" holds those random effects at zero, giving the typical-group curve. "sample" instead draws a new random effect from its estimated distribution, widening the uncertainty to include between-group variation. Set a seed with `set.seed()` for reproducible CIs. The available `by` values are `NULL` (a single population curve), "site", and `c("site", "year")`.

Value

A `kb_predictions` object: a summary tibble with `estimate`, `lower`, `upper`, the diameter predictor, and the `by` grouping columns.

See Also

[kb_predict_weight\(\)](#) for predictions at the rows of a supplied data frame.

Other prediction: [autoplot.kb_predictions\(\)](#), [kb_plot_predictions\(\)](#), [kb_predict_weight\(\)](#)

Examples

```
kb_predict_weight_by(fit_weight_sim_nereo, by = "site")
```

kb_prior_exponential	<i>Exponential Prior</i>
----------------------	--------------------------

Description

Construct an Exponential prior object for use in [kb_priors_weight_nereo\(\)](#) and the priors argument of the [kb_fit_*](#)() functions. Used for the standard deviation (scale) hyperparameters.

Usage

```
kb_prior_exponential(rate = 1)
```

Arguments

rate	A positive number giving the rate.
------	------------------------------------

Value

A `kb_prior_exponential` object.

See Also

Other priors: [kb_prior_normal\(\)](#), [kb_priors_weight_nereo\(\)](#)

Examples

```
kb_prior_exponential(rate = 1)
```

kb_prior_normal	<i>Normal Prior</i>
-----------------	---------------------

Description

Construct a Normal prior object for use in `kb_priors_weight_nereo()` and the `priors` argument of the `kb_fit_*`() functions.

Usage

```
kb_prior_normal(mean = 0, sd = 1)
```

Arguments

mean	A number giving the mean.
sd	A positive number giving the standard deviation.

Value

A `kb_prior_normal` object.

See Also

Other priors: `kb_prior_exponential()`, `kb_priors_weight_nereo()`

Examples

```
kb_prior_normal(mean = 0, sd = 2)
```

kb_priors_weight_nereo	<i>Default Priors for the Nereocystis Weight Model</i>
------------------------	--

Description

The default prior list for the *Nereocystis luetkeana* allometric weight model. Edit individual entries and pass the list to the `priors` argument of `kb_fit_weight_nereo()` to override defaults; unmodified entries keep their defaults.

Usage

```
kb_priors_weight_nereo()
```

Details

The prior family of each entry is fixed (the population-level terms are Normal, the standard deviations are Exponential); only the hyperparameters can be changed.

Value

A named list of prior objects with entries `intercept`, `diameter`, `diameter2`, `sd_site`, `sd_site_diameter`, `sd_site_year`, and `sd_residual`.

See Also

Other priors: [kb_prior_exponential\(\)](#), [kb_prior_normal\(\)](#)

Examples

```
priors <- kb_priors_weight_nereo()
priors$sd_site <- kb_prior_exponential(2)
```

 kb_stancode

Stan Source for a Model Fit

Description

The Stan source code of the fitted model, with comments stripped for a clean read. The raw source (comments included) is stored on the fit in `fit$meta$stancode`.

Usage

```
kb_stancode(fit, ...)

## S3 method for class 'kb_fit'
kb_stancode(fit, ...)
```

Arguments

<code>fit</code>	A <code>kb_fit</code> object.
<code>...</code>	Unused.

Value

A `kb_stancode` object: the comment-stripped Stan source string, which prints as readable code. Use [as.character\(\)](#) for the plain string.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
kb_stancode(fit_weight_sim_nereo)
```

log_lik.kb_fit	<i>Pointwise Log-Likelihood</i>
----------------	---------------------------------

Description

The pointwise log-likelihood of the observed data (from the Stan generated quantities), suitable for `loo::loo()`.

Usage

```
## S3 method for class 'kb_fit'
log_lik(object, ...)
```

Arguments

object	A kb_fit object.
...	Unused.

Value

A draws-by-observations (D x N) matrix.

See Also

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `posterior_epred.kb_fit_weight()`, `posterior_linpred.kb_fit_weight()`, `posterior_predict.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `residuals.kb_fit_weight()`, `samples()`, `summary.kb_fit()`, `tidy.kb_fit_weight()`

Examples

```
ll <- log_lik(fit_weight_sim_nereo)
dim(ll)
```

posterior_epred.kb_fit_weight	<i>Expected Weight Posterior Draws</i>
-------------------------------	--

Description

Draws from the expectation of the posterior predictive distribution (response-scale expected weight, exp of the linear predictor).

Usage

```
## S3 method for class 'kb_fit_weight'
posterior_epred(
  object,
  new_data = NULL,
  ...,
  new_levels = "sample",
  representative_site = NULL
)
```

Arguments

<code>object</code>	A <code>kb_fit_weight</code> object.
<code>new_data</code>	A data frame with a diameter column (and optional site / year columns), or NULL for the observed data.
<code>...</code>	Unused.
<code>new_levels</code>	A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with <code>set.seed()</code> for a reproducible interval.
<code>representative_site</code>	A character vector of site levels present in the fit, or NULL (the default). When supplied, a new or absent site takes its site main effects (intercept and slope) from the named reference site (the per-draw average when several are named), instead of the <code>new_levels</code> treatment; the <code>site:year</code> interaction still follows <code>new_levels</code> .

Details

Conditioning is inferred from the grouping columns present in `new_data`: a site (and optionally year) column with known levels is conditioned on; factors with no column are handled by `new_levels`. With `new_data = NULL` the observed data is used and conditioned on its site and year, so the central estimate agrees with `augment()`.

Value

A draws-by-observations ($D \times N$) matrix.

See Also

`kb_predict_weight()`, which summarises these draws.

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `log_lik.kb_fit()`, `posterior_linpred.kb_fit_weight()`, `posterior_predict.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `residuals.kb_fit_weight()`, `samples()`, `summary.kb_fit()`, `tidy.kb_fit_weight()`

Examples

```
ep <- posterior_epred(fit_weight_sim_nereo)
dim(ep)
```

```
posterior_linpred.kb_fit_weight
```

Linear-Predictor Posterior Draws

Description

Draws of the weight-model linear predictor on the log scale (or, with `transform = TRUE`, on the response scale).

Usage

```
## S3 method for class 'kb_fit_weight'
posterior_linpred(
  object,
  transform = FALSE,
  new_data = NULL,
  ...,
  new_levels = "sample",
  representative_site = NULL
)
```

Arguments

<code>object</code>	A <code>kb_fit_weight</code> object.
<code>transform</code>	A flag specifying whether to return the response-scale value (<code>exp</code>).
<code>new_data</code>	A data frame with a diameter column (and optional site / year columns), or NULL for the observed data.
<code>...</code>	Unused.
<code>new_levels</code>	A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with <code>set.seed()</code> for a reproducible interval.
<code>representative_site</code>	A character vector of site levels present in the fit, or NULL (the default). When supplied, a new or absent site takes its site main effects (intercept and slope) from the named reference site (the per-draw average when several are named), instead of the <code>new_levels</code> treatment; the <code>site:year</code> interaction still follows <code>new_levels</code> .

Details

Conditioning is inferred from the grouping columns present in `new_data` (see `posterior_epred()`); with `new_data = NULL` the observed data is used and conditioned on its site and year.

Value

A draws-by-observations (D x N) matrix.

See Also

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `log_lik.kb_fit()`, `posterior_epred.kb_fit_weight()`, `posterior_predict.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `residuals.kb_fit_weight()`, `samples()`, `summary.kb_fit()`, `tidy.kb_fit_weight()`

Examples

```
lp <- posterior_linpred(fit_weight_sim_nereo)
dim(lp)
```

posterior_predict.kb_fit_weight

Posterior-Predictive Weight Draws

Description

Draws from the posterior predictive distribution: the expected weight plus Student-t observation noise (scale `sWeight`, 4 degrees of freedom, matching the Stan likelihood). With `new_data = NULL` the stored yrep at the observed data is returned, for use with `bayesplot::pp_check()`.

Usage

```
## S3 method for class 'kb_fit_weight'
posterior_predict(
  object,
  new_data = NULL,
  ...,
  new_levels = "sample",
  representative_site = NULL
)
```

Arguments

<code>object</code>	A <code>kb_fit_weight</code> object.
<code>new_data</code>	A data frame with a diameter column (and optional site / year columns), or NULL for the stored yrep at the observed data.
<code>...</code>	Unused.

new_levels A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with `set.seed()` for a reproducible interval.

representative_site A character vector of site levels present in the fit, or NULL (the default). When supplied, a new or absent site takes its site main effects (intercept and slope) from the named reference site (the per-draw average when several are named), instead of the new_levels treatment; the site:year interaction still follows new_levels.

Details

For supplied new_data, conditioning is inferred from the grouping columns present (see `posterior_epred()`).

Value

A draws-by-observations (D x N) matrix.

See Also

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `log_lik.kb_fit()`, `posterior_epred.kb_fit_weight()`, `posterior_linpred.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `residuals.kb_fit_weight()`, `samples()`, `summary.kb_fit()`, `tidy.kb_fit_weight()`

Examples

```
pp <- posterior_predict(fit_weight_sim_nereo)
dim(pp)
```

predict.kb_fit_weight *Predict Method for a Weight Model Fit*

Description

A thin wrapper on `kb_predict_weight()`: predict weight at the supplied new_data rows (or the observed data when new_data = NULL). For allometric curves to visualise, use `kb_predict_weight_by()`.

Usage

```
## S3 method for class 'kb_fit_weight'
predict(
  object,
  new_data = NULL,
  ...,
  new_levels = c("sample", "average"),
  representative_site = NULL,
  conf_level = 0.95,
  estimate = stats::median,
  sig_fig = 3
)
```

Arguments

object	A kb_fit_weight object.
new_data	A data frame with a diameter column (and optional site / year columns), or NULL to predict at the observed data.
...	Unused.
new_levels	A string, one of "sample" or "average", controlling how random effects that are not conditioned on are treated (factors absent from the prediction, and any new level not seen in the fit). "sample" draws a new random effect from $\text{Normal}(0, \text{sd})$, widening the interval to include between-group variation; "average" holds the random effects at zero, giving the typical group. Known levels are always conditioned on. "sample" draws fresh randomness on each call, so set a seed with <code>set.seed()</code> for a reproducible interval.
representative_site	A character vector of site levels present in the fit, or NULL (the default). When supplied, a new or absent site takes its site main effects (intercept and slope) from the named reference site (the per-draw average when several are named), instead of the new_levels treatment; the site:year interaction still follows new_levels.
conf_level	A number between 0 and 1 giving the compatibility-interval level.
estimate	A function that reduces a numeric vector of posterior draws to a scalar point estimate (e.g. median or mean).
sig_fig	A whole number of significant figures for summary output.

Value

A kb_predictions object.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
predict(fit_weight_sim_nereo, data.frame(diameter = c(20, 40)))
```

prior_summary.kb_fit *Prior Summary*

Description

The resolved prior objects used to fit the model.

Usage

```
## S3 method for class 'kb_fit'
prior_summary(object, ...)
```

Arguments

object	A kb_fit object.
...	Unused.

Value

The named list of prior objects (see [kb_priors_weight_nereo\(\)](#)).

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
prior_summary(fit_weight_sim_nereo)
```

`residuals.kb_fit_weight`*Deviance Residuals*

Description

Posterior point estimates of the deviance residual at each observed row, from the Student-t log-weight likelihood, matching `augment()`'s residual column.

Usage

```
## S3 method for class 'kb_fit_weight'
residuals(object, ...)
```

Arguments

<code>object</code>	A <code>kb_fit_weight</code> object.
<code>...</code>	Unused.

Details

The deviance residual is computed per draw, then summarised with the posterior median.

Value

A numeric vector of deviance residuals, length `nobs(object)`.

See Also

`fitted()` for fitted values, and `augment()`.

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `log_lik.kb_fit()`, `posterior_epred.kb_fit_weight()`, `posterior_linpred.kb_fit_weight()`, `posterior_predict.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `samples()`, `summary.kb_fit()`, `tidy.kb_fit_weight()`

Examples

```
residuals(fit_weight_sim_nereo)
```

samples	<i>Posterior Draws</i>
---------	------------------------

Description

The raw posterior draws from a fitted model object.

Usage

```
samples(fit, ...)  
  
## S3 method for class 'kb_fit'  
samples(fit, ...)
```

Arguments

fit A fitted model object.
... Unused.

Value

A posterior draws object.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [summary.kb_fit\(\)](#), [tidy.kb_fit_weight\(\)](#)

Examples

```
samples(fit_weight_sim_nereo)
```

summary.kb_fit	<i>Summarise a Model Fit</i>
----------------	------------------------------

Description

A model fit’s metadata paired with a per-term posterior summary table.

Usage

```
## S3 method for class 'kb_fit'
summary(
  object,
  ...,
  conf_level = 0.95,
  estimate = stats::median,
  sig_fig = 3,
  include_random_effects = FALSE
)
```

Arguments

object	A kb_fit object.
...	Unused.
conf_level	A number between 0 and 1 giving the compatibility-interval level.
estimate	A function that reduces a numeric vector of posterior draws to a scalar point estimate (e.g. median or mean).
sig_fig	A whole number of significant figures for summary output.
include_random_effects	A flag specifying whether to include the group-level random-effect terms in the output.

Details

The print method renders a header (likelihood family, fixed- and random-effect structure, observation and group counts, sampler configuration, and the convergence verdict), the coefficient table, and a diagnostics footer. For a compact overview without the numeric table, call `print()` on the fit itself.

The coefficient table reports, per term:

estimate	the posterior point estimate (the estimate function; the median by default).
lower, upper	the conf_level equal-tailed compatibility limits.
rhat	the potential scale reduction factor, comparing between- and within-chain variance; values near 1 indicate convergence.
ess_bulk	the bulk effective sample size, governing the reliability of central posterior summaries.
ess_tail	the tail effective sample size, governing the reliability of the interval limits.

Population-level coefficients and random-effect standard deviations are always shown. The group-level deviations are included only when `include_random_effects = TRUE`, following the convention that `summary` reports the variance hyperparameters rather than the per-level effects (the latter are the `tidy()` default).

Value

A `summary_kb_fit` object: a list of fit metadata and a coefficients tibble with columns `term`, `estimate`, `lower`, `upper`, `rhat`, `ess_bulk`, and `ess_tail`.

See Also

Other generics: `augment.kb_fit()`, `coef.kb_fit()`, `converged.kb_fit()`, `fitted.kb_fit_weight()`, `glance.kb_fit()`, `kb_stancode()`, `log_lik.kb_fit()`, `posterior_epred.kb_fit_weight()`, `posterior_linpred.kb_fit_weight()`, `posterior_predict.kb_fit_weight()`, `predict.kb_fit_weight()`, `prior_summary.kb_fit()`, `residuals.kb_fit_weight()`, `samples()`, `tidy.kb_fit_weight()`

Examples

```
summary(fit_weight_sim_nereo)
```

tidy.kb_fit_weight	<i>Tidy a Weight Model Fit</i>
--------------------	--------------------------------

Description

Posterior summaries of the model terms: the population-level effects and the random-effect standard deviations, plus (when `include_random_effects = TRUE`) the group-level deviations. By default the per-level deviations are omitted, following the `broom.mixed` convention.

Usage

```
## S3 method for class 'kb_fit_weight'
tidy(
  x,
  ...,
  conf_level = 0.95,
  estimate = stats::median,
  sig_fig = 3,
  include_random_effects = FALSE
)
```

Arguments

<code>x</code>	A <code>kb_fit_weight</code> object.
<code>...</code>	Unused.
<code>conf_level</code>	A number between 0 and 1 giving the compatibility-interval level.
<code>estimate</code>	A function that reduces a numeric vector of posterior draws to a scalar point estimate (e.g. median or mean).
<code>sig_fig</code>	A whole number of significant figures for summary output.
<code>include_random_effects</code>	A flag specifying whether to include the group-level random-effect terms in the output.

Value

A tibble with one row per term and columns `term`, `estimate`, `lower`, `upper`.

See Also

Other generics: [augment.kb_fit\(\)](#), [coef.kb_fit\(\)](#), [converged.kb_fit\(\)](#), [fitted.kb_fit_weight\(\)](#), [glance.kb_fit\(\)](#), [kb_stancode\(\)](#), [log_lik.kb_fit\(\)](#), [posterior_epred.kb_fit_weight\(\)](#), [posterior_linpred.kb_fit_weight\(\)](#), [posterior_predict.kb_fit_weight\(\)](#), [predict.kb_fit_weight\(\)](#), [prior_summary.kb_fit\(\)](#), [residuals.kb_fit_weight\(\)](#), [samples\(\)](#), [summary.kb_fit\(\)](#)

Examples

```
tidy(fit_weight_sim_nereo)
```

Index

- * **datasets**
 - data_weight_sim_nereo, 6
 - fit_weight_sim_nereo, 6
- * **data**
 - data_weight_sim_nereo, 6
 - fit_weight_sim_nereo, 6
 - kb_check_data_weight_nereo, 9
- * **generics**
 - augment.kb_fit, 2
 - coef.kb_fit, 4
 - converged.kb_fit, 5
 - fitted.kb_fit_weight, 7
 - glance.kb_fit, 8
 - kb_stancode, 19
 - log_lik.kb_fit, 20
 - posterior_epred.kb_fit_weight, 20
 - posterior_linpred.kb_fit_weight, 22
 - posterior_predict.kb_fit_weight, 23
 - predict.kb_fit_weight, 24
 - prior_summary.kb_fit, 26
 - residuals.kb_fit_weight, 27
 - samples, 28
 - summary.kb_fit, 28
 - tidy.kb_fit_weight, 30
- * **model**
 - kb_fit_weight_nereo, 10
- * **prediction**
 - autoplot.kb_predictions, 3
 - kb_plot_predictions, 12
 - kb_predict_weight, 14
 - kb_predict_weight_by, 15
- * **priors**
 - kb_prior_exponential, 17
 - kb_prior_normal, 18
 - kb_priors_weight_nereo, 18

as.character(), 19

augment(), 7, 15, 21, 27

augment.kb_fit, 2

augment.kb_fit(), 4, 5, 7, 8, 19–21, 23–28, 30, 31

autoplot.kb_predictions, 3

autoplot.kb_predictions(), 13, 15, 17

coef.kb_fit, 4

coef.kb_fit(), 3, 5, 7, 8, 19–21, 23–28, 30, 31

converged(), 12

converged.kb_fit, 5

converged.kb_fit(), 3, 4, 7, 8, 19–21, 23–28, 30, 31

data_weight_sim_nereo, 6, 6, 7, 9

fit_weight_sim_nereo, 6, 6, 9

fitted(), 2, 3, 27

fitted.kb_fit_weight, 7

fitted.kb_fit_weight(), 3–5, 8, 19–21, 23–28, 30, 31

glance(), 12

glance.kb_fit, 8

glance.kb_fit(), 3–5, 7, 19–21, 23–28, 30, 31

kb_check_data_weight_nereo, 9

kb_check_data_weight_nereo(), 6, 7, 9, 11

kb_fit_progress, 10

kb_fit_progress(), 11, 12

kb_fit_weight_nereo, 10

kb_fit_weight_nereo(), 6, 7, 18

kb_plot_predictions, 12

kb_plot_predictions(), 3, 4, 15, 17

kb_predict_weight, 14

kb_predict_weight(), 4, 13, 17, 21, 24

kb_predict_weight_by, 15

kb_predict_weight_by(), 4, 13–15, 24

kb_prior_exponential, 17

kb_prior_exponential(), 18, 19

kb_prior_normal, 18
kb_prior_normal(), 17, 19
kb_priors_weight_nereo, 18
kb_priors_weight_nereo(), 11, 17, 18, 26
kb_stancode, 19
kb_stancode(), 3–5, 7, 8, 20, 21, 23–28, 30, 31

log_lik.kb_fit, 20
log_lik.kb_fit(), 3–5, 7, 8, 19, 21, 23–28, 30, 31

posterior_epred(), 7, 23, 24
posterior_epred.kb_fit_weight, 20
posterior_epred.kb_fit_weight(), 3–5, 7, 8, 19, 20, 23–28, 30, 31
posterior_linpred.kb_fit_weight, 22
posterior_linpred.kb_fit_weight(), 3–5, 7, 8, 19–21, 24–28, 30, 31
posterior_predict.kb_fit_weight, 23
posterior_predict.kb_fit_weight(), 3–5, 7, 8, 19–21, 23, 25–28, 30, 31
predict.kb_fit_weight, 24
predict.kb_fit_weight(), 3–5, 7, 8, 19–21, 23, 24, 26–28, 30, 31
prior_summary.kb_fit, 26
prior_summary.kb_fit(), 3–5, 7, 8, 19–21, 23–25, 27, 28, 30, 31

residuals(), 2, 3, 7
residuals.kb_fit_weight, 27
residuals.kb_fit_weight(), 3–5, 7, 8, 19–21, 23–26, 28, 30, 31
rstan::sampling(), 11
rstan::stan(), 11, 12

samples, 28
samples(), 3–5, 7, 8, 19–21, 23–27, 30, 31
summary(), 12
summary.kb_fit, 28
summary.kb_fit(), 3–5, 7, 8, 19–21, 23–28, 31

tidy(), 4
tidy.kb_fit_weight, 30
tidy.kb_fit_weight(), 3–5, 7, 8, 19–21, 23–28, 30