Package ‘artemis’

September 8, 2021

Title Analysis and Simulation of Environmental DNA Experiments

Version 1.1.1

Description Provides a modeling framework for the design and analysis of experiments collecting environmental DNA.

License GPL (>= 3)

Depends methods, graphics, stats, ggplot2, R (>= 3.5.0), loo

Imports Rcpp (>= 1.0.1), rstan (>= 2.19.0), lme4 (>= 1.1.21), rstantools (>= 2.1.1), utils

Encoding UTF-8

LazyData true

NeedsCompilation yes

SystemRequirements GNU make

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown, markdown, covr

VignetteBuilder knitr

Biarch true

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

Author Matt Espe [aut, cre], Myfanwy Johnston [aut]

Maintainer Matt Espe <mespe@ucdavis.edu>

Repository CRAN

Date/Publication 2021-09-08 19:10:09 UTC

R topics documented:

artemis-package ................................................................. 2
as.data.frame.eDNA_simulation ........................................... 3
conf_to_probs ................................................................. 3
Description

Simulating data and fitting models for eDNA studies can be accomplished using the artemis package. In particular, artemis implements a Bayesian latent-variable model in which the predictors affect a latent variable, eDNA concentration. This latent variable is related to an observed variable, CQ cycles via a standard curve calibration. CQ cycles, are often truncated at a certain limit, where more than X cycles is considered a non-detection.

Additional details on the model implemented by the artemis package can be found in the "Getting Started" vignette.

References

as.data.frame.eDNA_simulation

Methods for eDNA simulations

### Description

Methods for eDNA simulations

### Usage

```r
## S3 method for class 'eDNA_simulation'
as.data.frame(x, row.names, optional, ...)
```

### Arguments

- **x**: object of class eDNA_simulation
- **row.names**: ignored
- **optional**: ignored
- **...**: ignored

### Details

as.data.frame methods for eDNA simulations. This allows the conversion of the simulations to a form suitable for additional operations, e.g. plotting.

### Value

data.frame

### Author(s)

Matt Espe

---

conf_to_probs

Confidence to probability

### Description

Confidence level to probability

### Usage

```r
conf_to_probs(conf)
```
Arguments

conf numeric between 0 and 1

Details

This is a convenience function to convert from a confidence level to the lower and upper probability of the confidence interval.

Value

vector of length 2

Author(s)

Matt Espe

Examples

conf_to_probs(0.95)

cq_to_lnconc Convert CQ value to [eDNA]

Description

Convenience function to convert from Cq to [eDNA]

Usage

cq_to_lnconc(Cq_values, std_curve_alpha, std_curve_beta)

Arguments

Cq_values numeric vector, value of CQ
std_curve_alpha the alpha (intercept) value for the standard curve
std_curve_beta the beta (slope) value for the standard curve

Details

Convenience function to convert from Cq to [eDNA]

Value

numeric vector, [eDNA] values
Author(s)
Matt Espe

Description
Data from Delta Smelt (Hypomesus transpacificus) eDNA experiments, collected at the Central Valley Project facility in California.

Usage
eDNA_data

Format
A data frame of 180 observations and 9 variables associated with individual qPCR replicates within filters.

Date Date of sample, yyyy-mm-dd
FilterID Represents a single Sterivex filter. Each unique FilterID is associated with 3 qPCR reactions (technical replicates)
TechRep Index of technical replicate from a given SampleID.
Cq Quantification cycle of qPCR. For this assay, the limit of detection was set at 40.0
Distance_m Distance (in meters) from the 'source' of eDNA, which was a small underwater enclosure containing 100 live individual (cultured) Delta Smelt.
Volume_mL Volume (in milliliters) of water pulled through the Sterivex filter in the sample.
Biomass_N The number of individual Delta Smelt carcasses present in the submerged cage.
StdCrvAlpha_lnForm The intercept of the standard curve equation associated with these filters, in ln form
StdCrvBeta_lnForm The slope of the standard curve equation associated with these filters, in ln form

Details
This sample dataset is designed to be representative of the type of data collected in a semi-controlled eDNA survey study. The data consists of 180 technical replicates processed from samples filtered at different distances from the 'source' of eDNA, which was a small underwater enclosure containing 100 live individual (cultured) Delta Smelt. All samples were filtered from water flowing unidirectionally at the surface (depth < 1.0m). Filtered volume was fixed at either 50mL or 200mL. From distances of 10-50m, three filters were taken in series every 10m at 50mL and 200mL, sampled from near to far relative to live car. Note that the live car itself (Distance_m = ~0) was not actually sampled. Full details on this data are presented in Espe et al. 2021 (in prep).

The standard curve equation associated with this data is: Cq_star = -1.529*ln[eDNA concentration] + 21.168, where Cq_star is the Cq value estimated by the standard curve for the assay.
**Description**

Fit eDNA model

**Usage**

```r
eDNA_lm(
  formula,  
data,  
  std_curve_alpha,  
  std_curve_beta,  
  upper_Cq = 40,  
  prior_intercept = normal(location = -15, scale = 10),  
  priors = normal(),  
  ...
)
```

```r
eDNA_lmer(
  formula,  
data,  
  std_curve_alpha,  
  std_curve_beta,  
  upper_Cq = 40,  
  prior_intercept = normal(location = -15, scale = 10),  
  priors = normal(),  
  prior_random_variance = exponential(),  
  ...
)
```

```r
eDNA_zinf_lm(
  formula,  
data,  
  std_curve_alpha,  
  std_curve_beta,  
  upper_Cq = 40,  
  probability_zero = 0.08,  
  prior_intercept = normal(location = -15, scale = 10),  
  priors = normal(),  
  ...
)
```

```r
eDNA_zinf_lmer(
  formula,  
data,
```

```r
eDNA_zinf_lmer(
  formula,  
data,
```
std_curve_alpha,
std_curve_beta,
upper_Cq = 40,
probability_zero = 0.08,
prior_intercept = normal(location = -15, scale = 10),
priors = normal(),
prior_random_variance = exponential(),
...)

Arguments

formula a formula, specifying the relationship between the predictors and the latent variable eDNA concentration.
data data.frame, with the response and predictors
std_curve_alpha the alpha (intercept) value for the formula for converting between log(eDNA concentration) and CQ value
std_curve_beta the beta (slope) value for the formula for converting between log(eDNA concentration) and CQ value
upper_Cq numeric, the upper limit on CQ detection. Any value of log(concentration) which would result in a value greater than this limit is instead recorded as the limit.
prior_intercept named list such as created by rstanarm::normal. The list must contain elements named "location" and "scale", which are the location and scale for a normal prior over the intercept. Ignored when the intercept is omitted in the model formula.
priors named list such as created by rstanarm::normal. The list must contain elements named "location" and "scale", which are the location and scale for a normal prior over the betas, and "autoscale". If a single value is provided, this value will be repeated for each beta. If autoscale = TRUE, the scale of the priors is scaled by the sd of the predictors similar to rstanarm handles them.
... additional arguments passed to sampling
prior_random_variance the prior on variance of the random effects. Defaults to exponential distribution with rate 1.
probability_zero numeric, between 0 and 1. The probability of a non-detection from a source other than low concentration of eDNA, e.g. a filter failure. Defaults to 8 was the estimated p(zero) from a daily sampling experiment.

Details

These functions fit a Bayesian latent variable model to data collected from an eDNA sampling experiment. These data have a few particular characteristics that justify using a specialized model.
More details on these characteristics and the model structure, please refer to the "Getting Started" vignette for the artemis package.

There are four different modeling functions in the artemis package, eDNA_lm, eDNA_lmer, eDNA_zinf_lm, eDNA_zinf_lmer. eDNA_lm is for fitting a fixed effects model, while eDNA_lmer is for fitting a mixed or random effects model. The *_zinf versions implement a zero-inflated version of their respective lm function. All models are fit using the rstan::sampling function, which uses a Hamiltonian Monte Carlo algorithm to estimate parameters for the model. Users are encouraged to refer to the documentation for Stan and RStan at https://mc-stan.org/users/documentation/ for details about how models are fit.

Value

S4 object, with the following slots:

- **ln_conc** matrix, the posterior samples for the latent variable, eDNA concentration
- **Cq_star** matrix, the posterior prediction for the observed response
- **betas** array, the posterior estimates for the betas for the linear model
- **sigma_ln_eDNA** array, the posterior estimates for the measurement error of ln_eDNA
- **formula** formula, the original formula used in the model
- **x** data.frame, the model matrix used in the model
- **std_curve_alpha** numeric, the std. curve intercept value used
- **std_curve_beta** numeric, the std. curve slope value used
- **upper_Cq** numeric, the upper limit for observed CQ used
- **stanfit** stanfit, the original results from rstan::sampling

Diagnosing warning and error messages

The models have been written in Stan with key focus on robustness and speed. However, it is possible that users might encounter issues. Typically, these issues will be highlighted by warning messages coming from rstan::sampling. Often times, these warnings can be resolved by increasing the number of iterations that the HMC algorithm runs by specifying iters to be a larger value. This should be the first action attempted, as increasing the iters increases both the warm-up and sampling iterations. If users continue to have issues, additional control arguments can be passed to rstan::sampling via the ... argument.

Author(s)

Matt Espe

Examples

```r
## Fixed effect model
ans = eDNA_lm(Cq ~ Distance_m, eDNA_data,
               std_curve_alpha = 21.2, std_curve_beta = -1.5)
```
## Mixed-effect model

This takes a while to run

```r
ans2 = eDNA_lmer(Cq ~ Distance_m + (1|FilterID), eDNA_data,
                  std_curve_alpha = 21.2, std_curve_beta = -1.5)
```

---

### eDNA_model-class

**eDNA model fit results**

**Description**

An S4 object holding the results of a model fit by `eDNA_lm*`. 

**Slots**

- `ln_conc` matrix, the estimated latent variable, log(eDNA concentration)
- `Cq_star` matrix, the predicted CQ value for each obs
- `betas` matrix, the posterior estimate for each beta in the model
- `sigma_ln_eDNA` the estimated measurement error on ln_eDNA
- `formula` the formula used to fit the model
- `x` the model.matrix used to fit the model
- `std_curve_alpha` the alpha for the std. curve conversion formual used
- `std_curve_beta` the beta for the std. curve conversion formula used
- `upper_Cq` the upper limit for CQ
- `stanfit` the result from `rstan::sampling`
- `random_x` data.frame of the grouping variables used
- `random_sd` the estimated stdev. of each of the random effects

---

### eDNA_simulation-class

**eDNA simulation results**

**Description**

An S4 object holding the results of `sim_eDNA_lm*`. 

Slots

- \texttt{ln\_conc} matrix, simulated log(eDNA concentration)
- \texttt{Cq\_star} matrix, simulated CQ star
- \texttt{formula} the formula used
- \texttt{variable\_levels} the variable levels used
- \texttt{betas} the effect levels used
- \texttt{x} the model matrix used
- \texttt{std\_curve\_alpha} the alpha for the std. curve conversion formula used
- \texttt{std\_curve\_beta} the beta for the std. curve conversion formula used
- \texttt{upper\_Cq} the upper limit for CQ
- \texttt{groups} the grouping variables used
- \texttt{random\_sd} the stdev of the random effects

---

\texttt{est\_power\_lm} \hspace{2cm} \textit{Estimate power}

\subsection*{Description}

Estimate the power for a given eDNA design

\subsection*{Usage}

\begin{verbatim}
est\_power\_lm(
    formula,
    variable\_list,
    betas,
    sigma\_Cq,
    std\_curve\_alpha,
    std\_curve\_beta,
    type = c("exclude\_zero", "accuracy"),
    accuracy\_level = 0.2,
    conf\_level = 0.95,
    n\_sim = 200L,
    probs = conf\_to\_probs(conf\_level),
    upper\_Cq = 40,
    X = expand.grid(variable\_list),
    verbose = FALSE
)
est\_power\_lmer(
    formula,
    variable\_list,
    betas,
\end{verbatim}
est_power_ln

\[
\begin{align*}
\text{sigma}_\text{Cq}, \\
\text{sigma}_\text{rand}, \\
\text{std}_\text{curve}_\text{alpha}, \\
\text{std}_\text{curve}_\text{beta}, \\
type = c(\"exclude_zero\", \"accuracy\"), \\
\text{accuracy}_\text{level} = 0.2, \\
\text{conf}_\text{level} = 0.95, \\
n_\text{sim} = 200L, \\
\text{probs} = \text{conf}_\text{to}_\text{probs}(\text{conf}_\text{level}), \\
\text{upper}_\text{Cq} = 40, \\
X = \text{expand.grid}(\text{variable}_\text{list}), \\
\text{verbose} = \text{FALSE}
\end{align*}
\]

Arguments

- **formula**: a model formula, e.g. \(y \sim x_1 + x_2\). For \text{sim}_\text{eDNA}_\text{lmer}, random intercepts can also be provided, e.g. \((1 | \text{rep})\).
- **variable_list**: a named list, with the levels that each variable can take. Please note that the variables listed in the formula, including the response variable, must be present in the variable_list or in the X design matrix. Extra variables, i.e. variables which do not occur in the formula, are ignored.
- **betas**: numeric vector, the beta for each variable in the design matrix
- **sigma_Cq**: numeric, the measurement error on CQ.
- **std_curve_alpha**: the alpha value for the formula for converting between log(eDNA concentration) and CQ value
- **std_curve_beta**: the beta value for the formula for converting between log(eDNA concentration) and CQ value
- **type**: either "exclude_zero" or "accuracy". Exclude_zero give the classic power estimate, i.e. whether 0 is in the confidence interval for the estimate ("significant"). Accuracy measures whether the estimated betas are within some percentage of the "true" betas used to simulate the data.
- **accuracy_level**: numeric, between 0 and 1. The percent of the true betas for the accuracy estimate.
- **conf_level**: numeric, between 0 and 1, representing the percent of the confidence interval to calculate. If \text{probs} is not provided, then the interval is assumed to be symmetric.
- **n_sim**: integer, the number of simulations to conduct in order to estimate the power.
- **probs**: probabilities for the calculation of the confidence intervals. By default, a symmetric set of lower and upper probabilities is constructed by \text{conf}_\text{to}_\text{probs}.
- **upper_Cq**: numeric, the upper limit on CQ detection. Any value of log(concentration) which would result in a value greater than this limit is instead recorded as the limit.
- **X**: optional, a design matrix. By default, this is created from the variable_list using \text{expand.grid}(), which creates a balanced design matrix. However, the user
can provide their own X as well, in which case the variable_list is ignored. This allows users to provide an unbalanced design matrix.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>logical, when TRUE output from rstan::sampling is written to the console.</td>
</tr>
<tr>
<td>sigma_rand</td>
<td>numeric vector, the stdev for the random effects. There must be one sigma per random effect specified</td>
</tr>
</tbody>
</table>

**Details**

These functions estimate the power for a given eDNA design and specific effect sizes. The functions allow the user to specify to either use the classic definition of power (i.e., whether an estimate includes 0 in the confidence/credible interval), or accuracy, defined as the estimates being within a certain percentage of the "true" betas. Both can be useful in different circumstances, but eDNA survey studies produce data where accuracy can be a more useful metric. Since the response variable, Cq values, are truncated at an upper limit, it is often possible to detect a "significant effect" which results in Cq values above this upper limit, but the effects are estimated with a high amount of uncertainty. When are primarily interested in the precision of the estimate, the classic definition of power is not helpful. For these reasons, these functions allow the user to specify "accuracy" as the metric of interest.

**Value**

a named vector, with one estimate of the power for each parameter estimate.

**Author(s)**

Matt Espe

---

**est_power_range_lm**  Estimate power for a range of replicates

**Description**

Estimate power for range of reps.

**Usage**

```r
est_power_range_lm(
  formula,    
  variable_list, 
  betas, 
  sigma_Cq, 
  std_curve_alpha, 
  std_curve_beta, 
  type = c("exclude_zero", "accuracy"), 
  rep_range = seq(2, 20, 2), 
  accuracy_level = 0.2, 
  conf_level = 0.95, 
...)```
est_power_range_lm

n_sim = 200L,
probs = conf_to_probs(conf_level),
upper_Cq = 40,
verbose = FALSE
)

est_power_range_lmer(
  formula,
  variable_list,
  betas,
  sigma_Cq,
  sigma_rand,
  std_curve_alpha,
  std_curve_beta,
  type = c("exclude_zero", "accuracy"),
  rep_range = seq(2, 20, 2),
  accuracy_level = 0.2,
  conf_level = 0.95,
  n_sim = 200L,
  probs = conf_to_probs(conf_level),
  upper_Cq = 40,
  verbose = FALSE
)

Arguments

formula a model formula, e.g. \( y \sim x1 + x2 \). For sim_eDNA_lmer, random intercepts can also be provided, e.g. \( (1 | rep) \).

variable_list a named list, with the levels that each variable can take. Please note that the variables listed in the formula, including the response variable, must be present in the variable_list or in the X design matrix. Extra variables, i.e. variables which do not occur in the formula, are ignored.

betas numeric vector, the beta for each variable in the design matrix

sigma_Cq numeric, the measurement error on CQ.

std_curve_alpha the alpha value for the formula for converting between log(eDNA concentration) and CQ value

std_curve_beta the beta value for the formula for converting between log(eDNA concentration) and CQ value

type either "exclude_zero" or "accuracy". Exclude_zero give the classic power estimate, i.e. whether 0 is in the confidence interval for the estimate ("significant"). Accuracy measures whether the estimated betas are within some percentage of the "true" betas used to simulate the data.

rep_range vector, a set of the number of iterations to calculate the power for

accuracy_level numeric, between 0 and 1. The percent of the true betas for the accuracy estimate.
est_p_detect

conf_level numeric, between 0 and 1, representing the percent of the confidence interval to calculate. If probs is not provided, then the interval is assumed to be symmetric.

n_sim integer, the number of simulations to conduct in order to estimate the power.

probs probabilities for the calculation of the confidence intervals. By default, a symmetric set of lower and upper probabilities is constructed by conf_to_probs.

upper_Cq numeric, the upper limit on CQ detection. Any value of log(concentration) which would result in a value greater than this limit is instead recorded as the limit.

verbose logical, when TRUE output from rstan::sampling is written to the console.

sigma_rand numeric vector, the stdev for the random effects. There must be one sigma per random effect specified.

Details

This function estimates power for an eDNA sampling study for a range of potential reps.

Value

list, with each element a vector for the estimated power for each parameter for the corresponding number of replicates.

Author(s)

Matt Espe

---

est_p_detect Estimate the probability of detection

Description

Estimate the probability of detection

Usage

est_p_detect(
  variable_levels,
  betas,
  ln_eDNA_sd,
  std_curve_alpha,
  std_curve_beta,
  n_rep = 1:12,
  prob_zero = 0.08,
  model_fit = NULL,
  upper_Cq = 40
)
Arguments

variable_levels
numeric vector, with each element corresponding to the condition to estimate the probability of detection.

betas
numeric vector, the effect sizes for each of the variable level

ln_eDNA_sd
the measurement error on ln[DNA]. If a model_fit is provided and this is missing, the estimated sd(ln_eDNA) from the model will be used.

std_curve_alpha
the alpha for the std. curve formula for conversion between log(concentration) and CQ

std_curve_beta
the alpha for the std. curve formula for conversion between log(concentration) and CQ

n_rep
the number of replicate measurements at the levels specified

prob_zero
the probability of seeing a non-detection, i.e. zero, from a zero-inflated process. Defaults to 8 is the rate of inflated zeros in a large sampling experiment.

model_fit
optional, a model fit from eDNA_lm or eDNA_lmer. If this is provided, an estimate derived from the posterior estimates of beta is calculated.

upper_Cq
the upper limit on detection. Converted to the lower_bound of detection internally

Details

This function estimates the probability of getting a positive detection for an eDNA survey given a set of predictors. This can be useful when trying to take the estimates from a preliminary study and use those estimates to inform the deployment of future sampling schemes. The function assumes that you have either an idea of the effects of the various predictors, for example from a previous study, or a fit model with estimates of the effect sizes.

This function takes one circumstance at a time, and calculates the range of outcomes given a number of repeated sampling attempts. The probability calculated is the probability of getting at least one positive detection. For details on the underlying model and assumptions for this calculation, please refer to the package vignette.

Value

object of class "eDNA_p_detect" with the estimates of the probability of detection for the variable levels provided.

Notes on random effects

This function deals with random effects in two different ways. First, when we desire to see the probability of detection for a specific instance of a random effect, users can specify the random effect as just another effect by specifying the random effect = 1 in the variable list, and then the size of the random effect. However, when users wish to estimate the probability of detection in cases where random effects are generated from a distribution of random effects, this can be accomplished by adding the standard deviation of the random effect to the Cq_sd. This takes advantage of the fact that random effects are just another source of variation, and that sum of random normal distributions is itself a random normal distribution.
**Inconc_to_cq**

**Author(s)**
Matt Espe

**Examples**

```r
est_p_detect(variable_levels = c(Intercept = 1, Distance = 100, Volume = 20),
             betas = c(Intercept = -10.5, Distance = -0.05, Volume = 0.001),
             ln_eDNA_sd = 1, std_curve_alpha = 21.2, std_curve_beta = -1.5,
             n_rep = 1:12)
```

---

### lnconc_to_cq

*Convert [eDNA] to Cq*

**Description**

Convenience function for converting values

**Usage**

```r
lnconc_to_cq(x, std_curve_alpha, std_curve_beta, censor = TRUE, upper_Cq = 40)
```

**Arguments**

- `x` log[eDNA] values
- `std_curve_alpha` the alpha (intercept) value for the standard curve
- `std_curve_beta` the beta (slope) value for the standard curve
- `censor` logical, whether Cq values larger than the `upper_Cq` should be censored. If `TRUE`, these any values above the threshold will be replaced with the `upper_Cq` value.
- `upper_Cq` the max Cq value

**Details**

Convenience function for converting values

**Value**

vector of Cq values

**Author(s)**
Matt Espe
**Description**

A 'loo' method for `eDNA_model` objects, which is simply a wrapper around `loo(extract_log_lik(x))`

**Usage**

```r
## S3 method for class 'eDNA_model'
loo(x, ...)
```

**Arguments**

- `x`: a `eDNA_model` object
- `...`: additional args passed to `loo`

**Details**

See `?loo::loo` for more information about this calculation

**Value**

a list with the results of the calculation.

**Author(s)**

Matt Espe

---

**normal**

**Prior distributions**

**Description**

Normal prior distribution

**Usage**

```r
normal(location = 0, scale = 1, autoscale = TRUE)

exponential(rate = 1, autoscale = FALSE)
```
Arguments

- **location** numeric, the mean of the distribution
- **scale** numeric, the sd/scale of the distribution
- **autoscale** logical, whether the priors should be scaled. See ?rstanarm::priors for details on the scaling.
- **rate** numeric, the rate of the exponential distribution

Details

Parameters for the normal distribution, to be used for setting priors on model estimates. These are styled after the distributions in rstanarm.

Value

named list

Author(s)

Matt Espe

---

plot.eDNA_model  
*Plot eDNA model results*

Description

Plot eDNA model results

Usage

```r
## S3 method for class 'eDNA_model'
plot(x, y, pars = "betas", ...)
```

Arguments

- **x** object of class eDNA_model*
- **y** ignored
- **pars** parameters to plot
- **...** additional args passed to plot.stanfit

Details

Plot eDNA model results. Currently, this is just a wrapper for rstan::plot, which produces a "catapillar" plot.
Value

ggplot of posterior estimates

Author(s)

Matt Espe

Description

Plot the p(detect eDNA)

Usage

## S3 method for class 'eDNA_p_detect'
plot(
    x,
    y,
    probs = c(0.025, 0.975),
    ylim = c(0, 1),
    point_size = 2.5,
    n_breaks = 5,
    error_width = rel(1.25),
    ...  
)

Arguments

x object of class eDNA_p_detect
y ignored
probs probabilities for error bars when a model fit is supplied
ylim limits on the y axis
point_size point size for the estimated p(detect)
n_breaks passed to pretty
error_width the width of the ends of errorbars
... additional args passed to plot.stanfit

Details

This allows the visualization of the probability of detection produced by est_p_detect. If the object includes an interval estimate, errorbars are produced with those intervals.
Value

a ggplot object

Author(s)

Matt Espe

Description

Plot method for eDNA simulations

Usage

## S3 method for class 'eDNA_simulation'
plot(
  x,
  y,
  response = "Cq_star",
  probs = c(0.025, 0.975),
  alpha = 0.1,
  jitter_width = 0.35,
  ...
)

Arguments

x object of class eDNA_simulation
y ignored
response the response variable to plot
probs the probability for plotting CIs
alpha the alpha value, i.e. transparancy, of the points
jitter_width the width of the jitter applied to the points
... ignored

Details

Plot method for eDNA simulations which creates a separate plot of the simulated CQ values by each of the variables, i.e. a plot of the marginal distributions. Each plot is returned in a list of ggplot objects, each of which can be further augmented. By default, the points are semi-transparant and jittered to reduce overplotting.
Value

A list of ggplots

Author(s)

Matt Espe
Value

either a vector of predictions, or a matrix with the prediction plus interval, depending on the value of interval

Author(s)

Matt Espe

Arguments

x object of class "eDNA_model_*"
digits number of digits to show
... additional arguments passed to print
object an object of class eDNA_model

Details

Print method for eDNA model results. By default, this calls summary on the estimated effects, i.e. the "betas", and displays them with additional information about the model fit.

Value

x

Author(s)

Matt Espe
Description

Print method for p(detect)

Usage

```r
## S3 method for class 'eDNA_p_detect'
print(x, digits = getOption("digits"), ...)
```

Arguments

- `x`: object of class "eDNA_p_detect"
- `digits`: number of digits to show
- `...`: additional arguments passed to print

Details

Print method for the results of est_p_detect. If est_p_detect was provided with a single set of beta values, only these point estimates are printed. If a fit model was provided, posterior samples of the p(detect) are produced. These are summarized and printed together with posterior intervals, as estimated by the quantile method.

Value

- `x`

Author(s)

Matt Espe

Description

Print eDNA simulation
Usage

```r
## S3 method for class 'eDNA_simulation'
print(
x,
  FUN = summary,
  digits = getOption("digits"),
  show_variables = FALSE,
  ...
)
```

Arguments

- `x` object of class `eDNA_simulation`
- `FUN` a function to use to summarize the results, default is `summary`
- `digits` number of digits to show
- `show_variables` logical, should the variable levels used in the simulation be displayed. This is a bit redundant, since the variable levels are also displayed in the summary, but this can be helpful to check the results.
- `...` additional arguments passed to `print`

Details

Print method for the results of simulations of an eDNA experiment. By default, the results are summarized by each variable level using `summary`.

Value

`x`

Author(s)

Matt Espe

Description

Summarize random effects

Usage

```r
## S4 method for signature 'eDNA_model_lmer'
ranef(object, FUN = quantile, probs = c(0.025, 0.5, 0.975), ...)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an object of class eDNA_model</td>
</tr>
<tr>
<td>FUN</td>
<td>a posterior summary function, default quantile</td>
</tr>
<tr>
<td>probs</td>
<td>probabilities for the posterior summary function</td>
</tr>
<tr>
<td>...</td>
<td>extra args passed to FUN</td>
</tr>
</tbody>
</table>

Details

This function returns a summary of the random effects for an eDNA_model produced by eDNA_lmer.

Value

matrix, with one row per random effect, and one column per probability

Author(s)

Matt Espe

Description

Simulate eDNA data

Usage

```r
sim_eDNA_lm(
  formula,  # eDNA formula
  variable_list,  # variable names
  betas,  # parameter values
  sigma_ln_eDNA,  # noise parameter
  std_curve_alpha,  # standard curve parameters
  std_curve_beta,  # standard curve parameters
  n_sim = 1L,  # number of simulations
  upper_Cq = 40,  # upper Cq limit
  prob_zero = 0.08,  # probability of zero
  X = expand.grid(variable_list),  # design matrix
  verbose = FALSE  # print messages
)

sim_eDNA_lmer(
  formula,  # eDNA formula
  variable_list,  # variable names
  betas,  # parameter values
  sigma_ln_eDNA,  # noise parameter
  std_curve_alpha,  # standard curve parameters
  std_curve_beta,  # standard curve parameters
)```
sim_eDNA_lm

```r
sigma_rand,
std_curve_alpha,
std_curve_beta,
n_sim = 1L,
upper_Cq = 40,
prob_zero = 0.08,
X = expand.grid(variable_list),
verbose = FALSE
)
```

**Arguments**

- `formula` a model formula, e.g. `y ~ x1 + x2`. For `sim_eDNA_lmer`, random intercepts can also be provided, e.g. `( 1 | rep )`.
- `variable_list` a named list, with the levels that each variable can take. Please note that the variables listed in the formula, including the response variable, must be present in the `variable_list` or in the `X` design matrix. Extra variables, i.e. variables which do not occur in the formula, are ignored.
- `betas` numeric vector, the beta for each variable in the design matrix
- `sigma_ln_eDNA` numeric, the measurement error on ln[eDNA].
- `std_curve_alpha` the alpha value for the formula for converting between log(eDNA concentration) and CQ value
- `std_curve_beta` the beta value for the formula for converting between log(eDNA concentration) and CQ value
- `n_sim` integer, the number of cases to simulate
- `upper_Cq` numeric, the upper limit on CQ detection. Any value of log(concentration) which would result in a value greater than this limit is instead recorded as the limit.
- `prob_zero` numeric, between 0 and 1. The probability of seeing a non-detection (i.e., a "zero") via the zero-inflated mechanism. Defaults to 0.08.
- `X` optional, a design matrix. By default, this is created from the `variable_list` using `expand.grid()`, which creates a balanced design matrix. However, the user can provide their own `X` as well, in which case the `variable_list` is ignored. This allows users to provide an unbalanced design matrix.
- `verbose` logical, when TRUE output from `rstan::sampling` is written to the console.
- `sigma_rand` numeric vector, the stdev for the random effects. There must be one sigma per random effect specified

**Details**

These functions allow for computationally efficient simulation of Cq values from a hypothetical eDNA sampling experiment via a series of effect sizes (`betas`) on a number of predictor or variable levels (`variable_levels`). The mechanism for this model is described in detail in the artemis "Getting Started" vignette.
The simulation functions call to specialized functions which are written in Stan and are compiled to provide speed. This also allows the simulation functions and the modeling functions to reflect the same process at the code level.

Value

S4 object of class "eDNA_simulation_lm/lmer" with the following slots:

- **ln_conc** matrix the simulated log(concentration)
- **Cq_star** matrix the simulated CQ values, including the measurement error
- **formula** the formula for the simulation
- **variable_levels** named list, the variable levels used for the simulation
- **betas** numeric vector, the betas for the simulation
- **x** data.frame, the design matrix
- **std_curve_alpha** numeric the alpha for the std curve conversion
- **std_curve_beta** numeric the beta for the std curve conversion
- **upper_Cq** the upper limit for CQ

Diagnosing "unrealistic" simulations

Users will find that sometimes the simulationed response (i.e. Cq values) produced by this function are not similar to expected data collected from a sampling experiment. This circumstance suggests that there is a mismatch between the assumptions of the model and the data generating process in the field. For these circumstances, we suggest:

1. Check that the **betas** provided are the effect sizes on the predictor on the log[eDNA concentration], and not the Cq values.
2. Check that the variable levels provided are representative of real-world circumstances. For example, a sample volume of 0 ml is not possible.
3. Verify the values for the standard curve alpha and beta. These are specific to each calibration for the lab, so it is important that you use the same conversion between Cq values and log[eDNA concentration] as the comparison data.

Author(s)

Matt Espe

Examples

```r
## Includes extra variables
vars = list(Intercept = -10.6,
            distance = c(0, 15, 50),
            volume = c(25, 50),
            biomass = 100,
            alive = 1,
            tech_rep = 1:10,
            rep = 1:3, Cq = 1)
```
## Intercept only
ans = sim_eDNA_lm(Cq ~ 1, vars,
betas = c(intercept = -15),
sigma_ln_eDNA = 1e-5,
std_curve_alpha = 21.2, std_curve_beta = -1.5)

print(ans)

ans = sim_eDNA_lm(Cq ~ distance + volume, vars,
betas = c(intercept = -10.6, distance = -0.05, volume = 0.1),
sigma_ln_eDNA = 1, std_curve_alpha = 21.2, std_curve_beta = -1.5)

---

**summary.eDNA_model**  
**Summary of an eDNA model**

### Description
Summary of eDNA model

### Usage

```r
## S3 method for class 'eDNA_model'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

### Arguments

- **object**: an object of class eDNA_model
- **probs**: probabilities for the quantiles of the posterior samples
- **...**: currently ignored

### Details
Summary of eDNA model

### Value

data.frame, with summary of the fixed effects from the model

### Author(s)

Matt Espe
summary.eDNA_p_detect  

**Summary method for eDNA p(detect)**

**Description**

Summary method for eDNA p(detect)

**Usage**

```r
## S3 method for class 'eDNA_p_detect'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

**Arguments**

- `object`: an object of class `eDNA_p_detect`
- `probs`: probabilities for summary, passed to `quantile`
- `...`: ignored

**Details**

Summary method for eDNA p(detect)

**Value**

a data.frame, with summary statistics of the object

**Author(s)**

Matt Espe

summary.eDNA_simulation  

**Summary of eDNA simulations**

**Description**

Summary of eDNA simulations

**Usage**

```r
## S3 method for class 'eDNA_simulation'
summary(object, var = "Cq_star", probs = c(0.025, 0.5, 0.975), ...)
```
**Arguments**

- **object**: an object of class `eDNA_simulation_*`
- **var**: the simulated variable to summarize, either "Cq_star" or "ln_conc"
- **probs**: probabilities for the summary of the posterior samples
- **...**: currently ignored

**Details**

Summary of eDNA simulations

**Value**

data.frame

**Author(s)**

Matt Espe

---

**Description**

A 'waic' method for `eDNA_model` objects, which is simply a wrapper around `waic(extract_log_lik(x))`

**Usage**

```r
## S3 method for class 'eDNA_model'
waic(x, ...)
```

**Arguments**

- **x**: a `eDNA_model` object
- **...**: additional args passed to `waic`

**Details**

See `?loo::waic` for more information about this calculation

**Value**

a list with the results of the calculation.

**Author(s)**

Matt Espe
Index

* datasets
  eDNA_data, 5
  ANY-method (print.eDNA_model), 22
  artemis (artemis-package), 2
  artemis-package, 2
  as.data.frame.eDNA_simulation, 3
  conf_to_probs, 3
  cq_to_lnconc, 4
  eDNA_data, 5
  eDNA_lm, 6
  eDNA_lmer (eDNA_lm), 6
  eDNA_model, (print.eDNA_model), 22
  eDNA_model-class, 9
  eDNA_model_lmer-class (eDNA_model-class), 9
  eDNA_model_lmer-class (eDNA_model-class), 9
  eDNA_simulation-class, 9
  eDNA_simulation_lmer-class (eDNA_simulation-class), 9
  eDNA_simulation_lmer-class (eDNA_simulation-class), 9
  eDNA_zinf_lm (eDNA_lm), 6
  eDNA_zinf_lmer (eDNA_lm), 6
  est_p_detect, 14
  est_power_lm, 10
  est_power_lmer (est_power_lm), 10
  est_power_range_lm, 12
  est_power_range_lmer (est_power_range_lm), 12
  exponential (normal), 17
  lnconc_to_cq, 16
  loo.eDNA_model, 17
  normal, 17
  plot.eDNA_model, 18

plot.eDNA_p_detect, 19
plot.eDNA_simulation, 20
predict.eDNA_model, 21
predict.eDNA_model_lmer (predict.eDNA_model), 21
print.eDNA_model, 22
print.eDNA_p_detect, 23
print.eDNA_simulation, 23
ranef.eDNA_model_lmer-method, 24
sampling, 7
show.eDNA_model-method (print.eDNA_model), 22
sim_eDNA_lm, 25
sim_eDNA_lmer (sim_eDNA_lm), 25
summary.eDNA_model, 28
summary.eDNA_p_detect, 29
summary.eDNA_simulation, 29
waic.eDNA_model, 30

normal, 17
plot.eDNA_model, 18