Package 'eDNAjoint'

May 8, 2026

Title Joint Modeling of Traditional and Environmental DNA Survey Data in a Bayesian Framework

Version 0.3.3

Maintainer Abigail G. Keller <agkeller@berkeley.edu>

Description Models integrate environmental DNA (eDNA) detection data and traditional survey data to jointly estimate species catch rate (see package vignette: <https://ednajoint.netlify.app/>). Models can be used with count data via traditional survey methods (i.e., trapping, electrofishing, visual) and replicated eDNA detection/nondetection data via polymerase chain reaction (i.e., PCR or qPCR) from multiple survey locations. Estimated parameters include probability of a false positive eDNA detection, a site-level covariates that scale the sensitivity of eDNA surveys relative to traditional surveys, and gear scaling coefficients for traditional gear types. Models are implemented with a Bayesian framework (Markov chain Monte Carlo) using the 'Stan' probabilistic programming language.

License GPL-3

URL https://github.com/ropensci/eDNAjoint, https://docs.ropensci.org/eDNAjoint/

BugReports https://github.com/ropensci/eDNAjoint/issues

Encoding UTF-8

RoxygenNote 7.3.2

Biarch true

Depends R (>= 4.1)

Imports bayestestR, dplyr, ggplot2, lifecycle, loo, methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlist, rstan (>= 2.26.23), rstantools (>= 2.3.1.1), tidyr, scales

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.23), StanHeaders (>= 2.26.22)

SystemRequirements GNU make

LazyData true

Suggests bayesplot, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Abigail G. Keller [aut, cre],
Ryan P. Kelly [ctb],
Chitra M. Saraswati [rev],
Saras M. Windecker [rev]

Repository CRAN

Date/Publication 2025-06-21 02:30:05 UTC

Contents

eDNAjoint-package 2
detection_calculate 3
detection_plot 5
goby_data 7
green_crab_data 8
joint_model 9
joint_select 12
joint_summarize 14
mu_critical 15
traditional_model 16

Index **20**

`eDNAjoint-package` *The 'eDNAjoint' package.*

Description

Models integrate environmental DNA (eDNA) detection data and traditional survey data to jointly estimate species catch rate (see package vignette: https://ednajoint.netlify.app/). Models can be used with count data via traditional survey methods (i.e., trapping, electrofishing, visual) and replicated eDNA detection/nondetection data via polymerase chain reaction (i.e., PCR or qPCR) from multiple survey locations. Estimated parameters include probability of a false positive eDNA detection, a site-level covariates that scale the sensitivity of eDNA surveys relative to traditional surveys, and catchability coefficients for traditional gear types. Models are implemented with a Bayesian framework (Markov chain Monte Carlo) using the 'Stan' probabilistic programming language.

Author(s)

Maintainer: Abigail G. Keller <agkeller@berkeley.edu>

Other contributors:

- Ryan P. Kelly <rpkelly@uw.edu> [contributor]

- Chitra M. Saraswati [reviewer]
- Saras M. Windecker [reviewer]

References

Stan Development Team (NA). RStan: the R interface to Stan. https://mc-stan.org

See Also

Useful links:

- https://github.com/ropensci/eDNAjoint
- https://docs.ropensci.org/eDNAjoint/
- Report bugs at https://github.com/ropensci/eDNAjoint/issues

---

`detection_calculate` *Calculate the survey effort necessary to detect species presence, given the species expected catch rate.*

Description

This function calculates the number of survey effort units to necessary detect species presence using median estimated parameter values from joint_model(). Detecting species presence is defined as producing at least one true positive eDNA detection or catching at least one individual. See more examples in the Package Vignette.

Usage

```
detection_calculate(
  model_fit,
  mu,
  cov_val = NULL,
  probability = 0.9,
  pcr_n = 3
)
```

Arguments

`model_fit`	An object of class `stanfit`.
`mu`	A numeric vector of species densities/capture rates. If multiple traditional gear types are represented in the model, mu is the catch rate of gear type 1.
`cov_val`	A numeric vector indicating the values of site-level covariates to use for prediction. Default is NULL.
`probability`	A numeric value indicating the probability of detecting presence. The default is 0.9.
`pcr_n`	An integer indicating the number of PCR replicates per eDNA sample. The default is 3.

Value

A summary table of survey efforts necessary to detect species presence, given mu, for each survey type.

Note

Before fitting the model, this function checks to ensure that the function is possible given the inputs. These checks include:

- Input model fit is an object of class 'stanfit'.
- Input mu is a numeric vector.
- Input probability is a univariate numeric value.
- If model fit contains alpha, cov_val must be provided.
- Input cov_val is numeric.
- Input cov_val is the same length as the number of estimated covariates.
- Input model fit has converged (i.e. no divergent transitions after warm-up).

If any of these checks fail, the function returns an error message.

Examples

```
# Ex. 1: Calculating necessary effort for detection with site-level
# covariates

# Load data
data(goby_data)

# Fit a model including 'Filter_time' and 'Salinity' site-level covariates
fit_cov <- joint_model(data = goby_data, cov = c('Filter_time','Salinity'),
                       family = "poisson", p10_priors = c(1,20), q = FALSE,
                       multicore = FALSE)

# Calculate at the mean covariate values
# (covariates are standardized, so mean = 0)
detection_calculate(fit_cov$model, mu = seq(from = 0.1, to = 1, by = 0.1),
                    cov_val = c(0,0), pcr_n = 3)

# Calculate mu_critical at salinity 0.5 z-scores greater than the mean
detection_calculate(fit_cov$model, mu = seq(from = 0.1, to = 1, by = 0.1),
                    cov_val = c(0,0.5), pcr_n = 3)

# Ex. 2: Calculating necessary effort for detection with multiple traditional
# gear types

# Load data
data(green_crab_data)

# Fit a model with no site-level covariates
fit_q <- joint_model(data = green_crab_data, cov = NULL, family = "negbin",
                     p10_priors = c(1,20), q = TRUE, multicore = FALSE)
```

```
# Calculate
detection_calculate(fit_q$model, mu = seq(from = 0.1, to = 1, by = 0.1),
                    cov_val = NULL, pcr_n = 3)

# Change probability of detecting presence to 0.95
detection_calculate(fit_q$model, mu = 0.1, cov_val = NULL,
                    probability = 0.95, pcr_n = 3)
```

`detection_plot` *Plot the survey effort necessary to detect species presence, given the species expected catch rate.*

Description

This function plots the number of survey effort units to necessary detect species presence, calculated using median estimated parameter values from joint_model(). Detecting species presence is defined as producing at least one true positive eDNA detection or catching at least one individual. See more examples in the Package Vignette.

Usage

```
detection_plot(
  model_fit,
  mu_min,
  mu_max,
  cov_val = NULL,
  probability = 0.9,
  pcr_n = 3
)
```

Arguments

`model_fit`	An object of class `stanfit`.
`mu_min`	A value indicating the minimum expected species catch rate for plotting. If multiple traditional gear types are represented in the model, mu is the catch rate of gear type 1.
`mu_max`	A value indicating the maximum expected species catch rate for plotting. If multiple traditional gear types are represented in the model, mu is the catch rate of gear type 1.
`cov_val`	A numeric vector indicating the values of site-level covariates to use for prediction. Default is NULL.
`probability`	A numeric value indicating the probability of detecting presence. The default is 0.9.
`pcr_n`	An integer indicating the number of PCR replicates per eDNA sample. The default is 3.

Value

A plot displaying survey efforts necessary to detect species presence, given mu, for each survey type.

Note

Before fitting the model, this function checks to ensure that the function is possible given the inputs. These checks include:

- Input model fit is an object of class 'stanfit'.
- Input mu_min is a numeric value greater than 0.
- Input mu_max is a numeric value.
- If model fit contains alpha, cov_val must be provided.
- Input cov_val is numeric.
- Input cov_val is the same length as the number of estimated covariates.
- Input probability is a univariate numeric value.
- Input model fit has converged (i.e. no divergent transitions after warm-up).

If any of these checks fail, the function returns an error message.

Examples

```
# Ex. 1: Calculating necessary effort for detection with site-level
# covariates

# Load data
data(goby_data)

# Fit a model including 'Filter_time' and 'Salinity' site-level covariates
fit_cov <- joint_model(data = goby_data, cov = c('Filter_time','Salinity'),
                       family = "poisson", p10_priors = c(1,20), q = FALSE,
                       multicore = FALSE)

# Plot at the mean covariate values (covariates are standardized, so mean=0)
detection_plot(fit_cov$model, mu_min = 0.1, mu_max = 1,
               cov_val = c(0,0), pcr_n = 3)

# Calculate mu_critical at salinity 0.5 z-scores greater than the mean
detection_plot(fit_cov$model, mu_min = 0.1, mu_max = 1, cov_val = c(0,0.5),
               pcr_n = 3)

# Ex. 2: Calculating necessary effort for detection with multiple
# traditional gear types

# Load data
data(green_crab_data)

# Fit a model with no site-level covariates
fit_q <- joint_model(data = green_crab_data, cov = NULL, family = "negbin",
```

```
                    p10_priors = c(1,20), q = TRUE,
                    multicore = FALSE)

# Calculate
detection_plot(fit_q$model, mu_min = 0.1, mu_max = 1,
               cov_val = NULL, pcr_n = 3)

# Change probability of detecting presence to 0.95
detection_plot(fit_q$model, mu_min = 0.1, mu_max = 1, cov_val = NULL,
               probability = 0.95, pcr_n = 3)
```

---

goby_data *goby_data*

Description

goby_data

Usage

```
goby_data
```

Format

A list with four matrices representing eDNA sampling data (pcr_n and pcr_k), seine sampling data (count), and site-level covariate data (site_cov).

pcr_n Total number of eDNA qPCR replicates at each site (row) and eDNA sample replicate (column). Data includes 39 total sites and a maximum of 22 eDNA sample replicates. NA indicates that fewer eDNA samples were collected than the maximum at a site.

pcr_k Total number of positive eDNA qPCR detections at each site (row) and eDNA sample replicate (column). Data includes 39 total sites and a maximum of 22 eDNA sample replicates. NA indicates that fewer eDNA samples were collected than the maximum at a site.

count Count of goby individuals in seine samples at each site (row) and seine sample replicate (column). Data includes 39 total sites and a maximum of 22 seine replicates. NA indicates that fewer seine samples were collected than the maximum at a site.

site_cov Data representing site-level covariates at each site (row). Data includes mean salinity at a site ('Salinity'), mean time to filter eDNA samples ('Filter_time'), density of other fish species ('Other_fishes'), size of habitat ('Hab_size'), and presence of vegetation ('Veg'). All non-integer covariate data is standardized.

Source

https://datadryad.org/dataset/doi:10.5061/dryad.6rs23

References

Schmelzle, M.C. and Kinziger, A.P. (2016). Using occupancy modelling to compare environmental DNA to traditional field methods for regional-scale monitoring of an endangered aquatic species. Molecular Ecology Resources. 16(4): 895-908.

`green_crab_data` *green_crab_data*

Description

green_crab_data

Usage

```
green_crab_data
```

Format

A list with four matrices representing eDNA sampling data (pcr_n and pcr_k) and trap sampling data (count and count_type).

pcr_n Total number of eDNA qPCR replicates at each site (row) and eDNA sample replicate (column). Data includes 20 total sites and 5 eDNA sample replicates.

pcr_k Total number of positive eDNA qPCR detections at each site (row) and eDNA sample replicate (column). Data includes 20 total sites and 5 eDNA sample replicates.

count Count of green crab individuals in trap samples at each site (row) and trap sample replicate (column). Data includes 20 total sites and a maximum of 420 trap replicates. NA indicates that fewer trap samples were collected than the maximum at a site.

count_type Integer indicating the traditional gear type used at each site (row) and trap sample replicate (column). '1' refers to Fukui traps, and '2' refers to Minnow traps. Data includes 20 total sites and a maximum of 420 trap replicates. NA indicates that fewer trap samples were collected than the maximum at a site.

Source

doi:10.6084/m9.figshare.15117102.v2

References

Keller, A.G., Grason, E.W., McDonald, P.S., Ramon-Laca, A., Kelly, R.P. (2022). Tracking an invasion front with environmental DNA. Ecological Applications. 32(4): e2561. https://doi.org/10.1002/eap.2561

`joint_model` *Specify and fit joint model using count data from traditional surveys and eDNA PCR data*

Description

This function implements a Bayesian model that integrates data from paired eDNA and traditional surveys, as described in Keller et al. (2022) <doi.org/10.1002/eap.2561>. The model estimates parameters including the expected species catch rate and the probability of false positive eDNA detection. This function allows for optional model variations, like inclusion of site-level covariates that scale the sensitivity of eDNA sampling relative to traditional sampling, as well as estimation of gear scaling coefficients that scales the relative catchability of multiple traditional gear types. Model is implemented using Bayesian inference using the `rstan` package, which uses Hamiltonian Monte Carlo to simulate the posterior distributions. See more examples in the Package Vignette.

Usage

```
joint_model(
  data,
  cov = NULL,
  family = "poisson",
  p10_priors = c(1, 20),
  q = FALSE,
  phi_priors = NULL,
  multicore = FALSE,
  initial_values = NULL,
  n_chain = 4,
  n_warmup = 500,
  n_iter = 3000,
  thin = 1,
  adapt_delta = 0.9,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

`data`	A list containing data necessary for model fitting. Valid tags are `pcr_n`, `pcr_k`, `count`, `count_type`, and `site_cov`. `pcr_n` and `pcr_k` are matrices or data frames with first dimension equal to the number of sites (i) and second dimension equal to the maximum number of eDNA samples at a given site (m). `pcr_n` contains the total number of PCR replicates per site and eDNA sample, and `pcr_k` contains the total number of positive PCR detections per site and eDNA sample. `count` is a matrix or data frame of traditional survey count data, with first dimension equal to the number of sites (i) and second dimension equal to the maximum number of traditional survey replicates at a given site (j). `count_type`

	is an optional matrix or data frame of integers indicating gear type used in corresponding count data, with first dimension equal to the number of sites (i) and second dimension equal to the maximum number of traditional survey replicates at a given site. Values should be integers beginning with 1 (referring to the first gear type) to n (last gear type). `site_cov` is an optional matrix or data frame of site-level covariate data, with first dimension equal to the number of sites (i). `site_cov` should include informative column names. Empty cells should be NA and will be removed during processing. Sites, i, should be consistent in all PCR, count, and site covariate data.
`cov`	A character vector indicating the site-level covariates to include in the model. Default value is NULL.
`family`	The distribution class used to model traditional survey count data. Options include poisson ('poisson'), negative binomial ('negbin'), and gamma ('gamma'). Default value is 'poisson'.
`p10_priors`	A numeric vector indicating beta distribution hyperparameters (alpha, beta) used as the prior distribution for the eDNA false positive probability (p10). Default vector is c(1,20).
`q`	A logical value indicating whether to estimate gear scaling coefficients, q, for traditional survey gear types (TRUE) or to not estimate gear scaling coefficients, q, for traditional survey gear types (FALSE). Default value is FALSE.
`phi_priors`	A numeric vector indicating gamma distribution hyperparameters (shape, rate) used as the prior distribution for phi, the overdispersion in the negative binomial distribution for traditional survey gear data. Used when family = 'negbin.' If family = 'negbin', then default vector is c(0.25,0.25), otherwise, default is NULL.
`multicore`	A logical value indicating whether to parallelize chains with multiple cores. Default is FALSE.
`initial_values`	A list of lists of initial values to use in MCMC. The length should equal the number of MCMC chains. Initial values can be provided for parameters: beta, p10 (log-scale), mu, q, alpha. If no initial values are provided, default random values are drawn.
`n_chain`	Number of MCMC chains. Default value is 4.
`n_warmup`	A positive integer specifying the number of warm-up MCMC iterations. Default value is 500.
`n_iter`	A positive integer specifying the number of iterations for each chain (including warmup). Default value is 3000.
`thin`	A positive integer specifying the period for saving samples. Default value is 1.
`adapt_delta`	Numeric value between 0 and 1 indicating target average acceptance probability used in `rstan::sampling`. Default value is 0.9.
`verbose`	Logical value controlling the verbosity of output (i.e., warnings, messages, progress bar). Default is TRUE.
`seed`	A positive integer seed used for random number generation in MCMC. Default is NULL, which means the seed is generated from 1 to the maximum integer supported by R.

Value

A list of:

- a model object of class `stanfit` returned by `rstan::sampling`
- initial values used in MCMC

Note

Before fitting the model, this function checks to ensure that the model specification is possible given the data files. These checks include:

- All tags in data are valid (i.e., include pcr_n, pcr_k, count, count_type, and site_cov).
- Dimensions of pcr_n and pcr_k are equal, and dimensions of count and count_type are equal (if count_type provided).
- Number of sites in PCR and count data are equal.
- All data are numeric (i.e., integer or NA).
- Empty data cells (NA) match in pcr_n and pcr_k and in count and count_type.
- family is either 'poisson', 'negbin', or 'gamma'.
- p10_priors and phi_priors (if used) is a vector of two numeric values.
- site_cov has same number of rows as pcr_n and count, if present
- site_cov is numeric, if present
- cov values match column names in site_cov, if present

If any of these checks fail, the function returns an error message.

Examples

```
# Ex. 1: Implementing the joint model

# Load data
data(goby_data)

# Examine data in list
names(goby_data)

# Note that the surveyed sites (rows) should match in all data
dim(goby_data$pcr_n)[1]
dim(goby_data$count)[1]

# Fit a basic model with paired eDNA and traditional survey data.
# Count data is modeled using a poisson distribution.
fit <- joint_model(data = goby_data, family = "poisson",
                   p10_priors = c(1, 20),
                   multicore = FALSE)

# Ex. 2: Implementing the joint model with site-level covariates

# With the same data, fit a model including 'Filter_time' and 'Salinity'
```

```
# site-level covariates
# These covariates will scale the sensitivity of eDNA sampling relative to
# traditional surveys
# Count data is modeled using a poisson distribution.
fit_cov <- joint_model(data = goby_data, cov = c('Filter_time','Salinity'),
                       family = "poisson", p10_priors = c(1, 20),
                       multicore = FALSE)


# Ex. 3: Implementing the joint model with multiple traditional gear types

# Load data
data(green_crab_data)

# Examine data in list
names(green_crab_data)

# Note that the surveyed sites (rows) should match in all data
dim(green_crab_data$pcr_n)[1]
dim(green_crab_data$count)[1]

# Fit a model estimating a gear scaling coefficient for traditional survey
# gear types.
# This model does not assume all traditional survey methods have the same
# catchability.
# Count data is modeled using a negative binomial distribution.
fit_q <- joint_model(data = green_crab_data, cov = NULL, family = "negbin",
                     p10_priors = c(1, 20), q = TRUE,
                     phi_priors = c(0.25, 0.25),
                     multicore = FALSE, initial_values = NULL,
                     n_chain = 4, n_warmup = 500,
                     n_iter = 3000, thin = 1, adapt_delta = 0.9,
                     verbose = TRUE, seed = 123)
```

`joint_select` *Perform model selection using leave one out cross validation of model objects*

Description

This function performs leave one out cross validation of a list of model fits using functions in the `loo` package, as described in Vehtari, Gelman, and Gabry (2017) doi:10.1007/s11222-016-9696-4. Compare models fit using `joint_model()` or models fits using `traditional_model()`. See more examples in the Package Vignette.

Usage

```
joint_select(model_fits)
```

Arguments

model_fits	A list containing model fits of class stanfit.

Value

A matrix of delta elpd (expected log pointwise predictive density) between model fits. Function is performed using the loo package.

Note

Before model selection, this function makes the following check:

- Input is a list of model fits of class 'stanfit'.
- All models compared were fit wither either joint_model() or all with traditional_model().

If any of these checks fail, the function returns an error message.

Examples

```
data(green_crab_data)

# Fit a model without estimating a gear scaling coefficient for traditional
# survey gear types.
# This model assumes all traditional survey methods have the same
# catchability.
# Count data is modeled using a poisson distribution.
fit_no_q <- joint_model(data = green_crab_data, family = "poisson",
                        p10_priors = c(1,20), q = FALSE, multicore = FALSE)


# Fit a model estimating a gear scaling coefficient for traditional
# survey gear types.
# This model does not assume all traditional survey methods have the
# same catchability.
# Gear type 1 is used as the reference gear type.
# Count data is modeled using a negative binomial distribution.
fit_q <- joint_model(data = green_crab_data, family = "negbin",
                     p10_priors = c(1,20), q = TRUE, multicore = FALSE)

# Perform model selection
joint_select(model_fits = list(fit_no_q$model, fit_q$model))
```

`joint_summarize` *Summarize posterior distributions of model parameters.*

Description

This function summarizes the posterior distributions of specified parameters from a model fit. Summary includes mean, sd, and specified quantiles, as well as effective sample size (n_eff) and Rhat for estimated parameters. See more examples in the Package Vignette.

Usage

```
joint_summarize(model_fit, par = "all", probs = c(0.025, 0.975), digits = 3)
```

Arguments

`model_fit`	An object of class `stanfit`.
`par`	A character vector of parameter names. The default is 'all'.
`probs`	A numeric vector of quantiles of interest. The default is c(0.025,0.975).
`digits`	An integer indicating the number of decimal places to round values in summary table. Default value is 3.

Value

A summary table of parameter estimates.

Note

Before fitting the model, this function checks to ensure that the function is possible given the inputs. These checks include:

- Input model fit is an object of class 'stanfit'.
- Input probs is a numeric vector.
- Input par is a character vector.
- Input par are present in fitted model.
- Input model fit has converged (i.e. no divergent transitions after warm-up).

If any of these checks fail, the function returns an error message.

Examples

```
data(green_crab_data)

# Fit a model
model_fit <- joint_model(data = green_crab_data, family = "negbin", q = TRUE,
                         multicore = FALSE)

# Create summary table of all parameters
```

```
joint_summarize(model_fit$model)

# Summarize just 'p10' parameter
joint_summarize(model_fit$model, par = "p10", probs = c(0.025, 0.975),
                digits = 3)
```

`mu_critical` *Calculate mu_critical*

Description

This function uses the full posterior distributions of parameters estimated by `joint_model()` to calculate mu_critical, or the expected catch rate at which the probabilities of a false positive eDNA detection and true positive eDNA detection are equal. See more examples in the Package Vignette.

Usage

```
mu_critical(model_fit, cov_val = NULL, ci = 0.9)
```

Arguments

`model_fit`	An object of class `stanfit`
`cov_val`	A numeric vector indicating the values of site-level covariates to use for prediction. Default is NULL.
`ci`	Credible interval calculated using highest density interval (HDI). Default is 0.9 (i.e., 90% credible interval).

Value

A list with median mu_critical and lower and upper bounds on the credible interval. If multiple gear types are used, a table of mu_critical and lower and upper credible interval bounds is returned with one column for each gear type.

Note

Before fitting the model, this function checks to ensure that the function is possible given the inputs. These checks include:

- Input model fit is an object of class 'stanfit'.
- Input credible interval is a univariate numeric value greater than 0 and less than 1.
- Input model fit contains p10 parameter.
- If model fit contains alpha, cov_val must be provided.
- Input cov_val is numeric.
- Input cov_val is the same length as the number of estimated covariates.
- Input model fit has converged (i.e. no divergent transitions after warm-up).

If any of these checks fail, the function returns an error message.

Examples

```
# Ex. 1: Calculating mu_critical with site-level covariates

# Load data
data(goby_data)

# Fit a model including 'Filter_time' and 'Salinity' site-level covariates
fit_cov <- joint_model(data = goby_data, cov = c('Filter_time','Salinity'),
                       family = "poisson", p10_priors = c(1,20), q = FALSE,
                       multicore = FALSE)

# Calculate mu_critical at the mean covariate values (covariates are
# standardized, so mean = 0)
mu_critical(fit_cov$model, cov_val = c(0,0), ci = 0.9)

# Calculate mu_critical at habitat size 0.5 z-scores greater than the mean
mu_critical(fit_cov$model, cov_val = c(0,0.5), ci = 0.9)

# Ex. 2: Calculating mu_critical with multiple traditional gear types

# Load data
data(green_crab_data)

# Fit a model with no site-level covariates
fit_q <- joint_model(data = green_crab_data, cov = NULL, family = "negbin",
                     p10_priors = c(1,20), q = TRUE, multicore = FALSE)

# Calculate mu_critical
mu_critical(fit_q$model, cov_val = NULL, ci = 0.9)
```

`traditional_model` *Specify and fit model using count data from traditional, non eDNA surveys*

Description

This function implements a Bayesian model that estimates expected species catch rate using count data from traditional, non eDNA surveys. When multiple traditional gear types are used, an optional variation allows estimation of gear scaling coefficients, which scale the catchability of gear types relative to the expected catch rate of a reference gear type. Model is implemented using Bayesian inference using the `rstan` package, which uses Hamiltonian Monte Carlo to simulate the posterior distributions. See more examples in the Package Vignette.

Usage

```
traditional_model(
  data,
```

```
  family = "poisson",
  q = FALSE,
  phi_priors = NULL,
  multicore = FALSE,
  initial_values = NULL,
  n_chain = 4,
  n_warmup = 500,
  n_iter = 3000,
  thin = 1,
  adapt_delta = 0.9,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

data	A list containing data necessary for model fitting. Valid tags are count and count_type. count is a matrix or data frame of traditional survey count data, with first dimension equal to the number of sites (i) and second dimension equal to the maximum number of traditional survey replicates at a given site (j). count_type is an optional matrix or data frame of integers indicating gear type (k) used in corresponding count data, with first dimension equal to the number of sites (i) and second dimension equal to the maximum number of traditional survey replicates at a given site (j). Values should be integers beginning with 1 (referring to the first gear type) to n (last gear type). Empty cells should be NA and will be removed during processing. Sites, i, should be consistent in all count data.
family	The distribution class used to model traditional survey count data. Options include poisson ('poisson'), negative binomial ('negbin'), and gamma ('gamma'). Default value is 'poisson'.
q	A logical value indicating whether to estimate gear scaling coefficients, q, for traditional survey gear types (TRUE) or to not estimate gear scaling coefficients, q, for traditional survey gear types (FALSE). Default value is FALSE.
phi_priors	A numeric vector indicating gamma distribution hyperparameters (shape, rate) used as the prior distribution for phi, the overdispersion in the negative binomial distribution for traditional survey gear data. Used when family = 'negbin.' If family = 'negbin', then default vector is c(0.25,0.25), otherwise, default is NULL.
multicore	A logical value indicating whether to parallelize chains with multiple cores. Default is FALSE.
initial_values	A list of lists of initial values to use in MCMC. The length should equal the number of MCMC chains. Initial values can be provided for parameters: mu and q. If no initial values are provided, default random values are drawn.
n_chain	Number of MCMC chains. Default value is 4.
n_warmup	A positive integer specifying the number of warm-up MCMC iterations. Default value is 500.

n_iter	A positive integer specifying the number of iterations for each chain (including warmup). Default value is 3000.
thin	A positive integer specifying the period for saving samples. Default value is 1.
adapt_delta	Numeric value between 0 and 1 indicating target average acceptance probability used in rstan::sampling. Default value is 0.9.
verbose	Logical value controlling the verbosity of output (i.e., warnings, messages, progress bar). Default is TRUE.
seed	A positive integer seed used for random number generation in MCMC. Default is NULL, which means the seed is generated from 1 to the maximum integer supported by R.

Value

A list of:

- a model object of class stanfit returned by rstan::sampling
- initial values used in MCMC

Note

Before fitting the model, this function checks to ensure that the model specification is possible given the data files. These checks include:

- All tags in data are valid (i.e., include count and count_type).
- Number of sites in count and count type data are equal.
- All data are numeric (i.e., integer or NA).
- Empty data cells (NA) match in count and count_type.
- family is 'poisson', 'negbin', or 'gamma'.
- phi_priors (if used) is a vector of two numeric values.

If any of these checks fail, the function returns an error message.

Examples

```
# Load data
data(green_crab_data)

# Examine data in list
# This function uses only traditional survey count data and optionally
# the count type data
names(green_crab_data)

# Note that the surveyed sites (rows) should match in the data
dim(green_crab_data$count)[1]
dim(green_crab_data$count_type)[1]

# Fit a model without estimating a gear scaling coefficient for traditional
# survey gear types.
```

```
# This model assumes all traditional survey methods have the same
# catchability.
# Count data is modeled using a poisson distribution.
fit_no_q <- traditional_model(data = green_crab_data, family = "poisson",
                              q = FALSE, phi_priors = NULL,
                              multicore = FALSE, verbose = TRUE)


# Fit a model estimating a gear scaling coefficient for traditional survey
# gear types.
# This model does not assume all traditional survey methods have the same
# catchability.
# Count data is modeled using a negative binomial distribution.
fit_q <- traditional_model(data = green_crab_data, family = "negbin",
                           q = TRUE, phi_priors = c(0.25,0.25),
                           multicore = FALSE, initial_values = NULL,
                           n_chain = 4, n_warmup = 500, n_iter = 3000,
                           thin = 1, adapt_delta = 0.9, verbose = TRUE,
                           seed = 123)
```

Index

∗ **data**
goby_data, 7
green_crab_data, 8

detection_calculate, 3
detection_plot, 5

eDNAjoint *(*eDNAjoint-package*)*, 2
eDNAjoint-package, 2

goby_data, 7
green_crab_data, 8

joint_model, 9
joint_select, 12
joint_summarize, 14

mu_critical, 15

traditional_model, 16