

Package: bycatch (via r-universe)

October 18, 2026

Type Package

Title Fit Bayesian Models to Bycatch Data

Version 1.0.10

Maintainer Eric Ward <eric.ward@noaa.gov>

Description Fits models to bycatch data, and can expand to fleet-wide estimates, like in situations when fisheries are observed with less than 100% observer coverage.

License GPL-3

Encoding UTF-8

Depends R (>= 4.3.0)

Imports cli, ggplot2, methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang (>= 0.4.1), rstan (>= 2.18.1), rstantools (>= 2.4.0)

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)

Suggests testthat (>= 3.0.0), parallel, knitr, rmarkdown, devtools, MASS

URL <https://ericward-noaa.github.io/bycatch/>

BugReports <https://github.com/ericward-noaa/bycatch/issues>

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

SystemRequirements GNU make

Biarch true

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make

Repository <https://noaa-fisheries-integrated-toolbox.r-universe.dev>

Date/Publication 2026-09-03 00:33:05 UTC

RemoteUrl <https://github.com/ericward-noaa/bycatch>
RemoteRef HEAD
RemoteSha 52615d336ef4d72ebdc0dacb9363ffc31bc6b643

Contents

bycatch-package	2
extract_log_lik_for_loo	2
fit_bycatch	3
get_expanded	6
get_fitted	7
get_stream_summary	8
get_total	9
plot_expanded	10
plot_fitted	11
plot_pred_distribution	12
Index	14

bycatch-package	<i>The 'bycatch' package.</i>
-----------------	-------------------------------

Description

A DESCRIPTION OF THE PACKAGE

References

Stan Development Team (2023). RStan: the R interface to Stan. R package version 2.21.8.
<https://mc-stan.org>

extract_log_lik_for_loo	<i>Extract log-likelihood matrix for loo, removing NA observations</i>
-------------------------	--

Description

This function extracts the log-likelihood array from a fitted bycatch model and removes any columns (observations) that have NA values in any MCMC iteration. This is necessary because loo::loo() cannot handle NA values.

Usage

extract_log_lik_for_loo(fit)

Arguments

`fit` A bycatch model fit object (output from `fit_bycatch` or `fit_bycatch_modified`)

Value

A matrix suitable for `loo::loo()` with dimensions (n_iterations x n_valid_observations)

Examples

```
# Create example data
d <- data.frame(
  Year = 2002:2014,
  Takes = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  CovRate = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  Sets = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)

# Fit a model
fit <- fit_bycatch(Takes ~ 1,
  data = d,
  time = "Year",
  effort = "Sets",
  covrate = "CovRate",
  family = "poisson",
  time_varying = FALSE
)

# Extract log-likelihood for L00-CV
log_lik <- extract_log_lik_for_loo(fit)

# Run L00-CV
# library(loo)
# loo_result <- loo(log_lik)
```

fit_bycatch

fit_bycatch - Unified version with coverage rate parameters

Description

fit_bycatch - Unified version with coverage rate parameters

Usage

```
fit_bycatch(
  formula,
  data,
  time = "year",
  effort = "effort",
```

```

covrate = NULL,
expansion_rate = NULL,
takes_em = NULL,
effort_em = NULL,
covrate_em = NULL,
takes_both = NULL,
effort_both = NULL,
covrate_obs = NULL,
covrate_both = NULL,
effort_total = NULL,
family = c("poisson", "nbinom2", "poisson-hurdle", "nbinom2-hurdle", "lognormal",
           "gamma", "lognormal-hurdle", "gamma-hurdle", "normal", "normal-hurdle"),
time_varying = FALSE,
iter = 1000,
chains = 3,
control = list(adapt_delta = 0.9, max_treedepth = 20),
...
)
```

Arguments

formula	The model formula.
data	A data frame.
time	Named column of the 'data' data frame with the label for the time (e.g. year) variable
effort	Named column of the 'effort' variable in the data frame with the label for the fishing effort to be used in estimation of mean bycatch rate. This represents total observed effort
covrate	Optional: Column name for coverage rate (percentage) for single-stream models. Replaces expansion_rate.
expansion_rate	DEPRECATED: Use 'covrate' or 'covrate_obs' instead. The expansion rate to be used in generating distributions for unobserved sets.
takes_em	Optional: Column name for bycatch takes recorded by EM (electronic monitoring). Default NULL.
effort_em	Optional: Column name for effort monitored by EM. Required if takes_em is provided.
covrate_em	Optional: Column name for EM coverage rate (percentage). Used to calculate unobserved effort.
takes_both	Optional: Column name for bycatch takes when both Observer and EM are present. Default NULL.
effort_both	Optional: Column name for effort with both Observer and EM monitoring. Required if takes_both is provided.
covrate_obs	Optional: Column name for Observer coverage rate (percentage). Used to calculate unobserved effort.
covrate_both	Optional: Column name for "Both" coverage rate (percentage). Used to calculate unobserved effort.

effort_total	Optional: Column name for total fishery effort. If provided, takes precedence over coverage rates. Must be in same units as effort columns.
family	Family for response distribution can be discrete ("poisson", "nbinom2", "poisson-hurdle", "nbinom2-hurdle"), or continuous ("normal", "gamma", "lognormal", "normal-hurdle", "gamma-hurdle", "lognormal-hurdle"). The default distribution is "poisson". The hurdle variants estimate the probability of zeros (theta) separately from the other models and use truncated distribution to model positive counts. All use a log link function.
time_varying	boolean TRUE/FALSE, whether to include time varying component (this is a random walk, analogous to making this a Dynamic linear model)
iter	the number of mcmc iterations, defaults to 1000
chains	the number of mcmc chains, defaults to 3
control	List to pass to <code>rstan::sampling()</code> . For example, increase adapt_delta if there are warnings about divergent transitions: <code>control = list(adapt_delta = 0.99)</code> . By default, bycatch sets <code>adapt_delta = 0.9</code> .
...	Any other arguments to pass to <code>rstan::sampling()</code> .

Details

Coverage Rates vs effort_total:

This function supports two approaches for calculating unobserved effort:

1. **Coverage rates** (recommended): Use `covrate`, `covrate_obs`, `covrate_em`, `covrate_both` to specify what percentage of the fishery is monitored. The function calculates unobserved effort as: $\text{observed_effort} * ((100 - \text{coverage}) / \text{coverage})$
2. **Total effort**: Use `effort_total` to provide the total fishery effort directly. Unobserved effort is calculated as: $\text{effort_total} - \text{observed_effort}$ NOTE: `effort_total` must be in the same units as your effort columns

Priority order: `effort_total` > coverage rates > 100% coverage assumed

Multi-stream monitoring: When using multiple monitoring streams (Observer, EM, Both), the data are kept separate in the statistical model but all share the same underlying bycatch rate. This approach assumes all monitoring types have perfect/equal detection ($\text{detection} = 100\%$) but avoids distributional issues that arise from pooling.

Value

list of the data used to fit the model, the matrix of covariates, the expanded bycatch generated via the fit and simulations, and the fitted stan model

Examples

```
# Single stream example with coverage rate
d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "CovRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
```

```

    "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
  )
  fit <- fit_bycatch(Takes ~ 1,
    data = d, time = "Year",
    effort = "Sets",
    covrate = "CovRate",
    family = "poisson",
    time_varying = FALSE
  )

  # Multi-stream example with coverage rates (Observer + EM)
  d_multi <- data.frame(
    Year = 2002:2014,
    Takes_obs = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0),
    Sets_obs = c(200, 180, 150, 350, 250, 270, 180, 150, 400, 350, 280, 180, 250),
    CovRate_obs = c(20, 19, 17, 19, 21, 21, 20, 20, 20, 19, 20, 19, 19),
    Takes_em = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
    Sets_em = c(150, 120, 140, 250, 180, 190, 120, 100, 300, 270, 200, 130, 190),
    CovRate_em = c(15, 13, 16, 14, 15, 15, 13, 13, 15, 15, 14, 14, 15)
  )

  fit_multi <- fit_bycatch(Takes_obs ~ 1,
    data = d_multi,
    time = "Year",
    effort = "Sets_obs",
    covrate_obs = "CovRate_obs",
    takes_em = "Takes_em",
    effort_em = "Sets_em",
    covrate_em = "CovRate_em",
    family = "poisson"
  )

```

get_expanded

get_expanded is a helper function to return a matrix of posterior predictive values for unobserved bycatch

Description

get_expanded is a helper function to return a matrix of posterior predictive values for unobserved bycatch

Usage

```
get_expanded(fitted_model)
```

Arguments

fitted_model Data and fitted model returned from estimation

Value

matrix (MCMC draws x time steps) of posterior predictive values for unobserved bycatch

Examples

```
d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets",
  family = "poisson",
  expansion_rate = "expansionRate",
  time_varying = FALSE
)
expanded <- get_expanded(fit)
```

get_fitted	<i>get_fitted returns df of observed bycatch estimates (lambda of Poisson), accounting for effort but not accounting for observer coverage</i>
------------	--

Description

get_fitted returns df of observed bycatch estimates (lambda of Poisson), accounting for effort but not accounting for observer coverage

Usage

```
get_fitted(fitted_model, alpha = 0.05)
```

Arguments

fitted_model	Data and fitted model returned from fit_bycatch(). If a hurdle model, then the plot returns the total bycatch rate (including zero and non-zero components).
alpha	The alpha level for the credible interval, defaults to 0.05

Value

plot called from ggplot

Examples

```
d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year", effort = "Sets",
  family = "poisson", time_varying = FALSE
)
get_fitted(fit)
```

get_stream_summary	<i>get_stream_summary provides a summary of monitoring coverage and bycatch by stream</i>
--------------------	---

Description

get_stream_summary provides a summary of monitoring coverage and bycatch by stream

Usage

```
get_stream_summary(fitted_model, alpha = 0.05)
```

Arguments

fitted_model	Data and fitted model returned from fit_bycatch() with multi-stream monitoring
alpha	The alpha level for the credible interval, defaults to 0.05

Value

data frame with summary statistics for each monitoring stream

Examples

```
d <- data.frame(
  Year = 2002:2014,
  Takes_obs = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0),
  Sets_obs = c(200, 180, 150, 350, 250, 270, 180, 150, 400, 350, 280, 180, 250),
  Takes_em = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
  Sets_em = c(150, 120, 140, 250, 180, 190, 120, 100, 300, 270, 200, 130, 190),
  Sets_total = c(1000, 950, 900, 1800, 1200, 1300, 900, 750, 2000, 1800, 1400, 950, 1300)
)

fit <- fit_bycatch(Takes_obs ~ 1,
  data = d, time = "Year",
  effort = "Sets_obs",
```

```

    takes_em = "Takes_em",
    effort_em = "Sets_em",
    effort_total = "Sets_total",
    family = "poisson"
  )

  get_stream_summary(fit)

```

get_total

get_total is a helper function to return a matrix of total estimated bycatch

Description

get_total is a helper function to return a matrix of total estimated bycatch

Usage

```
get_total(fitted_model)
```

Arguments

fitted_model Data and fitted model returned from estimation

Value

matrix (MCMC draws x time steps) of posterior predictive values for total bycatch (observed + unobserved)

Examples

```

d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets",
  family = "poisson",
  expansion_rate = "expansionRate",
  time_varying = FALSE
)
total <- get_total(fit)

# Calculate total bycatch summaries
total_mean <- colMeans(total)
total_quantiles <- apply(total, 2, quantile, probs = c(0.025, 0.975))

```

plot_expanded	<i>plot_expanded is makes plots of the expanded bycatch estimates, accounting for observer coverage and effort</i>
---------------	--

Description

plot_expanded is makes plots of the expanded bycatch estimates, accounting for observer coverage and effort

Usage

```
plot_expanded(
  fitted_model,
  xlab = "Time",
  ylab = "Events",
  show_total = TRUE,
  include_points = FALSE
)
```

Arguments

fitted_model	Data and fitted model returned from estimation
xlab	X-axis label for plot
ylab	Y-axis label for plot
show_total	Whether to show the total predicted bycatch (by default, this is TRUE) or just the expanded unobserved events (=FALSE)
include_points	whether or not to include raw bycatch events on plots, defaults to FALSE

Value

plot called from ggplot

Examples

```
d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets",
  family = "poisson",
  expansion_rate = "expansionRate",
  time_varying = FALSE
)
```

```
plot_expanded(
  fitted_model = fit,
  xlab = "Year",
  ylab = "Fleet-level bycatch",
  include_points = TRUE
)
```

plot_fitted	<i>plot_fitted makes plots bycatch estimates (lambda of Poisson), accounting for effort but not accounting for observer coverage</i>
-------------	--

Description

plot_fitted makes plots bycatch estimates (lambda of Poisson), accounting for effort but not accounting for observer coverage

Usage

```
plot_fitted(
  fitted_model,
  xlab = "Time",
  ylab = "Events",
  include_points = FALSE,
  alpha = 0.05
)
```

Arguments

fitted_model	Data and fitted model returned from fit_bycatch(). If a hurdle model, then only then the plot returns the total bycatch rate (including zero and non-zero components).
xlab	X-axis label for plot
ylab	Y-axis label for plot
include_points	whether or not to include raw bycatch events on plots, defaults to FALSE
alpha	The alpha level for the credible interval, defaults to 0.05

Value

plot called from ggplot

Examples

```
d <- data.frame(
  "Year" = 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)
)
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year", effort = "Sets",
  family = "poisson", time_varying = FALSE
)
plot_fitted(fit,
  xlab = "Year", ylab = "Fleet-level bycatch",
  include_points = TRUE
)

# fit a negative binomial model, with more chains and control arguments
fit_nb <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets", family = "nbinom2",
  time_varying = FALSE, iter = 2000, chains = 4,
  control = list(adapt_delta = 0.99, max_treedepth = 20)
)

# fit a time varying model
fit <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets", family = "poisson", time_varying = TRUE
)

# include data for expansion to unobserved sets
fit_nb <- fit_bycatch(Takes ~ 1,
  data = d, time = "Year",
  effort = "Sets", family = "nbinom2",
  expansion_rate = "expansionRate",
  time_varying = FALSE, iter = 2000, chains = 4,
  control = list(adapt_delta = 0.99, max_treedepth = 20)
)
```

plot_pred_distribution

plot_pred_distribution plots the distribution of bycatch prediction for a specific year

Description

plot_pred_distribution plots the distribution of bycatch prediction for a specific year

Usage

```
plot_pred_distribution(  
  fitted_model,  
  year,  
  xlab = "Predicted Bycatch",  
  ylab = "Frequency",  
  breaks = NULL  
)
```

Arguments

<code>fitted_model</code>	Data and fitted model returned from <code>fit_bycatch()</code> .
<code>year</code>	Specified year for the prediction distribution
<code>xlab</code>	X-axis label for plot
<code>ylab</code>	Y-axis label for plot
<code>breaks</code>	Breaks for bars in the plot, defaults to 1

Value

A ggplot object.

Examples

```
d <- data.frame(  
  "Year" = 2002:2014,  
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),  
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30, 7, 26, 21, 22, 23, 27),  
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486)  
)  
fit <- fit_bycatch(Takes ~ 1,  
  data = d, time = "Year",  
  effort = "Sets",  
  family = "poisson",  
  expansion_rate = "expansionRate",  
  time_varying = FALSE  
)  
plot_pred_distribution(fit, year = 2002)
```

Index

`bycatch (bycatch-package)`, [2](#)

`bycatch-package`, [2](#)

`extract_log_lik_for_loo`, [2](#)

`fit_bycatch`, [3](#)

`get_expanded`, [6](#)

`get_fitted`, [7](#)

`get_stream_summary`, [8](#)

`get_total`, [9](#)

`plot_expanded`, [10](#)

`plot_fitted`, [11](#)

`plot_pred_distribution`, [12](#)

`rstan::sampling()`, [5](#)