Bycatch Estimator User Guide

Elizabeth A. Babcock

2021-07-09

ebabcock@rsmas.miami.edus

Introduction

The R code called 2.BycatchModels.r runs a generic model-based bycatch estimation procedure, after you first set up the data inputs and other specifications in 1.BycatchModelSpecificationsExample.r. The code can estimate both total bycatch, calculated by expanding a sample, such as an observer database, to total effort from logbooks or landings records, and an annual index of abundance, calculated only from the observer data.

The code runs best in R studio. Before running the code for the first time, install the latest versions of R (R Core Team (2020)) and RStudio (RStudio Team (2020)), as well as the following libraries: tidyverse, ggplot2, MASS, lme4, cplm, tweedie, DHARMa, tidyselect, MuMIn, gridExtra, pdftools, foreach, doParallel, reshape2, and glmmTMB (Wickham et al. (2019); Wickham (2016); Venables and Ripley (2002); Bates et al. (2015); Zhang (2013); Dunn and Smyth (2005); Hartig (2020); Henry and Wickham (2020); Barton (2020); Auguie (2017); Wickham and Pedersen (2019); Iannone, Cheng, and Schloerke (2020); Ooms (2020);Microsoft and Weston (2020);Corporation and Weston (2020); Wickham (2007); Brooks et al. (2017)). The output figures and tables are printed to a pdf file using R Markdown and the knitr library (Xie (2015)), which outputs a LaTex file; therefore, you must have a LaTex programm installed, such as TinyTex (Xie (2019), Xie (2021)).

The R code estimates total bycatch as follows. First, mean catch per unit effort (CPUE) of observed sample units (trips in this example, but it could be sets) is estimated from a linear model with predictor variables. The observation error models used are delta-lognormal, delta-gamma, negative binomial (from either glm.nb in the MASS library or glmmTMB, nbinom1 and nbinom2) and Tweedie (from cpglm or glmmTMB). Within each observation error model group, potential predictor variables are chosen based on the user's choice of information criteria (AICc, AIC or BIC) (Barton (2020)). The user specifies a most complex and simplest model, and all intermediate models are considered. The user-specified simplest model will usually include year, and can also include, for example, stratification variables that are used in the observer program sampling design. The model with the lowest value of the information criterion is chosen as best within each observation error group.

The best candidate models in each observation error group are then compared using 10-fold cross-validation, if desired, to see which observation error model best predicts CPUE. The best model according to cross validation is the one with the lowest root mean square error (RMSE) in the predicted CPUE, excluding from consideration models that do not fit well according to criteria described below. Note that this model selection using information criteria and cross-validation is only intended as a guide. The user should also look at the information criteria across multiple models, residuals and other diagnostics, and may want to choose a different model for bycatch estimation or abundance index calculation based on other criteria, such as the design of the observer sampling program.

For the best model in each observation error model group, the total bycatch is estimated by predicting the catch in all logbook trips (i.e., the whole fishery) from the fitted model and summing across trips. The catch in each trip is predicted directly by the negative binomial models. Tweedie models predict CPUE, which is then multiplied by effort. Delta-lognormal and delta-gamma models have separate components for the probability of a positive CPUE and the CPUE, which must be multiplied together (with appropriate bias corrections) and multiplied by effort to get the total catch. Catch in each trip is summed across trips to get the total catch in each year. Because catch is being predicted in each trip, the variance of the prediction is calculated as the variance of the prediction interval, which is the standard error of the prediction squared plus the residual variance. Variances are summed across trips to get the total variance of the predictions in each year.

If the logbook data is aggregated across multiple trips (e.g. by strata) the effort is allocated equally to all the trips in a row of the logbook data table for the purpose of simulating catches.This allocation procedure is not needed to estimate the mean total bycatch, but it is necessary to estimate the variances correctly.

The model can estimate bycatch and abundance indices for multiple species or dispositons (e.g. dead discard, live release) from the same fishery simultaneously if they are all being estimated from the same data sets.

If a user requests an annual abundance index, this is calculated from the same models as were selected for bycatch estimation.

Data specification and model set up

The user must specify the names of the databases, the predictor variables to include (via the simplest and most complex model), and several other specifications in the file called 1.BycatchModelSpecificationExample.r. Everything else works automatically.

The observer data should be aggregated to the appropriate sample unit, either trips or sets. Effort must be in the same units in both data sets (e.g. sets or hook-hours). The logbook data may be aggregated to sample units, or it may be aggregated further, as long as it includes data on all stratification or predictor variables. For example, the data can be aggregated by year, region and season if those are the stratification variables, or it can be aggregated by trip. If any environmental variables, such as depth, are included, the logbook data probably has to be entered at the set level. The observer data should have columns for year and the other predictor variables, the observed effort and the observed bycatch or catch per trip of each species to be estimated. The logbook data must also have year and the other predictor variables, and the total effort in the same units (e.g. sets or hook-hours) as the observer data. There should also be a column that reports how many sample units (i.e. trips) are included in each row in the logbook data if the data are aggregated. This is needed to predict catches by trip for the variance calculations. If there are any NA values in any of the variables, those rows will be deleted from the data set.

Throughout the data specification file, change the values on the right hand side of the assignment arrow, but do not change the variable names on the left hand side. The first section specifies the data tables to be used.

```
#Specify directory where R files are found.
baseDir<-"C:/Users/ebabcock/Dropbox/bycatch project/Current R code"
setwd(baseDir)

#Give a name to the run, which will be used to set up a directory for the the outputs
runName<-"Simulated data example"

# What would you like to estimate?
# You may calculate either an annual abundance index, or total bycatch, or both
# If you want total bycatch, you must have logbook data or some other source of total effort
EstimateIndex<-TRUE
EstimateBycatch<-TRUE

#### Read in the observer data file. This could also be an assignment to an
```

```
# R object if you have already got the data in R.
obsdat<-read.csv("ExampleObs.csv",as.is=TRUE)

#### Specify name of file with total effort data, if you are estimating total bycatch.
# It can be aggregated or may include one line per sample unit (trip).
# If estimateByatch is FALSE, this variable is not needed.
logdat<-read.csv("ExampleLog.csv",as.is=TRUE)
```

Next, give the names of the variables in the observer and (if estimating bycatch) logbook data files. You must also specify the common and scientific names of the species being analyzed, the units of the bycatch estimates (e.g. kg, numbers), and the type of catch (e.g. retained catch, dead discards, live releases). If analyzing more than one species or disposition type, some of these inputs must be vectors with the same length as the number of species and/or disposition types.

Give the formulas for the most complex and simplest model to be considered. If the simplest model requires stratification variables other than year, summaries of the predicted bycatch at the level of these stratification variables will be printed to .csv files, but will not be plotted automatically. Abundance indices will be calculated including all the variables requested in indexVars, to allow for different indices for different stratification variables if desired (e.g. different spatial areas).

```
### What is the sample unit in this data set? e.g. sets or trips.
sampleUnit<-"trips" #Usually trips or sets

#Specify the name of the effort variable in the observer data and logbook data. These must be
#in the same units. (e.g. 1000 hook hours). Also specify a column for effort
#that is not sampled, in trips with observers. This can be zero in all cases if observers
#sample 100% of effort in sampled trips.
obsEffort<-"sampled.sets"
#obsEffortNotSampled<-"unsampled.sets"  #Not needed yet. May be added to later version
logEffort<-"sets" #This variable is only needed if estimating bycatch

# Give the name of the column in the logbook data that gives the number of sample units (trips or sets)
# that each row includes. If the logbook data is not aggregated (i.e. each row is a #sample unit) just
logNum<-NA

#Give common and scientific names. Can be a vector of names to do multiple species at the same time
#in which case each species must have its own column in obsdat.
#This example takes the columns from a data frame to run multiple species at once.
common<-"Simulated species"
sp<-"Genus species"

#Give the name of the columns associated with the species. If it is a vector,
# it must match the common and scientific names given above
obsCatch<-"Catch"

# Give units and type of catch to go in plot labels. Must be a vector of the same length as sp
catchUnit<-rep("number",length(sp))
catchType<-rep("dead discard", length(sp))
#Specify the name of the variable defining the Years in both databases
yearVar<-"Year"

# Specify the most complex and simplest model to be considered. The code will find compare all
# intermediate models using information criteria. Use indexModel to specify which strata to
# keep separate in calculating abundance indices.
```

```
complexModel<-formula(y~(Year+season)^2)
simpleModel<-formula(y~Year)
indexModel<-formula(y~Year)

## The variables must have identical names and factor levels in the observer and logbook data sets
#Specify which of these variables should be interpreted as categorical to make sure they are in factor
#Variables not in this list will retain their original format
factorNames=c("Year","season")
```

Next, specify which models to try and which information criterion to use in narrowing down the predictor variables to use in each observation error model group. Model selection is done with dredge function in the MuMIn library(Barton (2020)). Note that the tweedie outputs should be the same whether using TMB (TweedieTMB) or the cpglm function (Tweedie). The negative binomial 2 in TMB is the same as the classic negative binomial in glm.nb. To get faster results, use TMB only for these distributions. They are both included for comparison.

Also, specify whether to do cross-validation to choose between observation error models, and whether to use the dredge function to use information criteria to choose the best set of predictor variables for each fold in cross validation. If DredgeCrossValidation is FALSE, the same predictor variables will be used for each fold, as selected for the full data set. This saves time with large data sets. The variable ResidualsTest allows excluding from cross-validation any model where the residuals have a P<0.01 for a Kolmogorov Smirnov test of whether the residuals are distributed as expected under the likelihood (testUniformity in DHARMa). This is useful for excluding poorly performing models. However, it may be too restrictive if none of the models perform well.

Also, specify whether to save the R workspace after the models run. If the variable useParallel is true and your computer has multiple cores, the dredge function will be run in parallel. This greatly speeds up the calculations. If you have trouble getting this to work, set useParallel to FALSE.

```
#Specify which observation error models to try. Options are delta-lognormal, delta-gamma, negative
#binomial and tweedie, specified as: "Lognormal" for delta lognormal,"Gamma" for delta gamm,"NegBin" fo
#using glm.mb in the MASS library, "Tweedie" for cpglm, and TMB nbinom1, nbinom2, and tweedie in the gl
#library, specified with "TMB" followed by the model type. Binomial is run
#automatically as part of the delta models if either of them are selected.
modelTry<-c("Lognormal","TMBnbinom2","TMBtweedie")

#Specify preferred information criteria for model selection
# Choices are AICc, AIC and BIC.
selectCriteria<-"BIC"

#Specify whether to run a 10 fold cross-validation (TRUE or FALSE). This may not work with a small
#or unbalanced data set. DredgeCrossValidation specifies whether to use information criteria
#to find the best model in cross validation, using the dredge function, or just keep the same model for
#Do not use dredge for very large data sets, as the run will be slow.
DoCrossValidation<-TRUE
DredgeCrossValidation<-FALSE

#Specify whether to exclude models that fail the DHARMa residuals test.
ResidualTest<-FALSE

# Specify whether to save R workspace. This should be true unless you
# don't have space on your disk. Also specify whether to use parallel processing to
# speed up calculations.
saveR<-TRUE
useParallel<-TRUE
```

Finally, if you have information on total bycatch in each year to validate your estimates, for example in a simulation study, fill out the following. Otherwise, set plotValidation to FALSE.

```
## Validation. If you have true values of the total bycatch (for example in a simulation study)  Make P
plotValidation<-FALSE
trueVals<-NULL
trueCols<-NULL
```

Running the model

Once the specification file is set up, open the file 2.ByCatchModels.r. The first section allows you to input the name of the data specification file, loads the specifications and then loads a file called "4.PreliminarySetup.r" to load all the libraries, set up the data files, and print a preliminary data summary.

The data summaries are output to directories named "output" followed by the specified run name. A pdf file with summaries for all species (DataSummary.pdf) is placed in the main output folder, and a csv file for each species is placed in an output file named for the species. Note that records with NA in the catch or effort variable are excluded from the analysis and not included in the estimated sample size. If there are any years with no data, or no positive observations, you may want to exclude those years from the analysis. The summary table also counts the number of outliers (defined as data points more than 8 standard deviations from the mean) because outliers cause problems with fitting in some models. For data-checking, this output file also includes columns with estimated bycatch and its variance using a simple unstratified ratio estimator by year (See 3.BycatchFunctions.r for all the functions). You will get an error message if if any of the variables in the specified models are not found in the data frame.

```
# This code runs a generic model-based bycatch estimation procedure.
# The observer data should be aggregated to the appropriate sample unit, either trips or sets.
# Effort must be in the same units in both data sets (e.g. hook hours), but it does not matter
# how the logbook effort data is aggregated, as long has it includes data on all stratification or
# predictor variables. FOr example, the data can be aggregated by year, East-West and season if those
# are the stratification variables, or it can be aggregated by trip.
# The user must specify the names of the databases and the variables to include in the
# The .r file named on line 12. Everything else should work automatically.
# Contact Beth Babcock ebabcock@rsmas.miami.edu for assistance.

############### Step 1. Enter the data specification in the file named here ############################
specFile<-"C:/Users/ebabcock/Dropbox/bycatch project/Current R code/1.BycatchModelSpecificationExample.r
# Either set the working directory or put the full patch in the filename.
# Complete the information in the file before continuing. You may run through specFile line by line, bu
# will also be sourced again later. The file will be saved, with the addition of the date, to the outpu

##################################################################################
### From here on no changes should be needed.
#################################################################################
###############################################################################
#Read in data specification
#source(specFile) #Commented out for user guide
#Make data summary outputs
source(paste0(baseDir,"/4.preliminarySetup.r"))
# Stop here and check data summaries to make sure each species has reasonable number
# of observations in each year for analysis.
#The file called  "Data summary all species.pdf" has all the tables.
#There is also a .csv for each species.
```

Main analysis loop

The next section runs the loop (across the specified species, disposition types, etc.) to run the CPUE models, estimate total bycatch and/or abundance indices, and do cross-validation if requested. You can ignore the warnings and information about the models as long as the loop keeps running. Most of these are information about which variables are being tried and which models have converged, which will be summarized in the output files.

The model may be slow if you have a large data set or are doing cross-validation. The outputs all go into the directories labeled with the species names, and include a pdf will all results, and separate csv files for all the tables. Note, for this demo, we set run=1 to do one step at a time. When running the R code, this all runs in a loop. When the models are fit, the code keeps track of whether the model converged correctly, or if not, where it went wrong. An output table called modelFail.csv summarizes the results, with a "-" for models that converged successfully, "data" for models that could not be fit due to insufficient data (no positive observations in some year prevents fitting the delta models), "fit" for models that failed to converge, "cv" for models that produced results with unreasonably high CVs (>10) in the annual catch predictions, and "resid" for models that failed the Kolmogorov Smirnoff test for having the correct distribution in the DHARMa library (Hartig (2020)), if residualTest is TRUE. Models that fail in any of these ways are discarded and not used in cross-validation. If you get one of these errors for a model you want to use, you should check the data for missing combinations of predictor variables, extreme outliers, or years with too few positive observations.

The first section of the loop includes two nested loops that go through all the models in modelTry and find the best model using the specified information criterion. The first loop runs all the models that are applied to all data together (i.e. not delta models). The delta models are run separately in the second loop because they require a different data setup. There is a function called findBestModelFunc, which applies the dredge function from the MuMin library to find the best combination of the predictor variables according to the specified information criteria (Barton (2020)). The dredge function produces a table which includes all the information criteria for each model that was considered, as well as model weights calculated for the information criterion the user specified, which sum to one and indicate the degree of support for the model in the data. The best model will have the highest weight. But, in some cases other models with also have strong support, and should perhaps be considered, particularly if they are simpler. The current version of the code does not use MuMIn's model averaging function, but this may be worth considering if several models have similar weights.

A binomial model will be tried if it was requested, or if either a delta-lognormal or delta-gamma model were requested, since delta models have a binomial component. The binomial models are fitted using the glm function with a logit link. The negative binomial is run using the glm.nb function from the MASS library (Venables and Ripley (2002)) or nbinom1 nbinom2 from the glmmTMB library(Brooks et al. (2017)). The glm.nb function is very similar to the nbinom2 method in glmTMB, but both are included for comparison. Both define the variance of the negative binomial as:

$$\sigma^2 = \mu + \mu^2/\theta$$

, where θ is an estimated parameter. For nbinom1, the variance is defined as:

$$\sigma^2 = \mu(1+\alpha)$$

, where α is an estimated parameter. This version of the negative binomial model, which is equivalent to a quasi-Poisson model, gives somewhat different results from the other negative binomial models.

The negative binomial predicts integer counts, so it is appropriate for predicting bycatch in numbers per trip for all the trips in the logbook data. To allow this model to also be used with catch or bycatch measured in weights, the code rounds the catches to integers before running this model. Check that this is appropriate for the units you are using. To predict CPUE it is necessary to include an offset in the model. We use a log link for all three binomial models, so that the model predicts:

$$log(C_i) = b_0 + b_1 x_1 + offset(log(E_i))$$

were C_i is the catch in trip i in the observer data, $b_0 + b_1 x_1$ is an example linear predictor with an intercept and a slope, and the offset is the log of the effort E_i in each trip. This is algebraically equivalent to modeling CPUE as a function of the same linear predictor (without the offset).

The Tweedie distribution is available using the cpglm function in the cplm library (Zhang (2013)) or the tweedie family in glmmTMB (Brooks et al. (2017)) by including "Tweedie" or TMBtweedie in modelTry. The Tweedie is a generalized function that estimates a distribution similar to a gamma distribution, except that it allows extra probability mass at zero. It is thus appropriate for continuous data with extra zeros. It uses a log link, and, in addition to the linear predictor for the log(mean) it estimates an index parameter p and dispersion parameter ϕ which together determine the shape of the distribution.

If all years have at least one positive observation, the loop next runs the lognormal and gamma models if requested. For the lognormal model, the CPUE is log transformed, and the mean CPUE for positive observations is modeled with the lm function. For the gamma method, the default inverse link is used to model the positive CPUE values, using the glm function.

```
############## This is the main analysis loop ##########################
StartTime<-Sys.time()
#for(run in 1:numSp) {
run<-1  #For user guide, set run to 1 instead of looping
 datval<-dat[[run]]
 outVal<-dirname[[run]]
 varExclude<-NULL
 #Fit all models except delta
 for(mod in which(!modelTry %in% c("Lognormal","Gamma"))){
   modfit1<-findBestModelFunc(datval,modelTry[mod],printOutput=TRUE)
   modelSelectTable[[run]][[modelTry[mod]]]<-modfit1[[2]]
   modFits[[run]][[modelTry[mod]]]<-modfit1[[1]]
 }
#Fit delta models
 if("Lognormal" %in% modelTry | "Gamma" %in% modelTry) {  #Delta models if requested
   posdat<-filter(dat[[run]],pres==1)
   y<-unlist(lapply(posdat[,factorNames],function(x) length(setdiff(levels(x),x)))) #See if all levels
   varExclude<-names(y)[y>0]
   if(length(varExclude>0)) print(paste(common[run], "excluding variable",varExclude,"from delta models
   if((min(summary(posdat$Year))>0 |  is.numeric(datval$Year)) &!is.null(modFits[[run]][["Binomial"]]))
     for(mod in which(modelTry %in% c("Lognormal","Gamma")))  {
       modfit1<-findBestModelFunc(posdat,modelTry[mod],printOutput=TRUE)
       modelSelectTable[[run]][[modelTry[mod]]]<-modfit1[[2]]
       modFits[[run]][[modelTry[mod]]]<-modfit1[[1]]
     }
   } else {
     print("Not all years have positive observations, skipping delta models")
     modelFail[run,c("Lognormal","Gamma")]<-"data"
     modPredVals[[run]][[modelTry[mod]]]<-NULL
     modIndexVals[[run]][[modelTry[mod]]]<-NULL
   }
  }
```

The next section loops through all the models again, and calculates the model residuals, residual diagnostics from the DHARMa library (Hartig (2020)), and, if requested, calculations of the total bycatch and an index of abundance.

For the best model (according to the information criterion) in each model group, both ordinary residuals and DHARMa scaled residuals are plotted, and the DHARMa diagnostics are calculated. The DHARMa library uses simulation to generate scaled residuals based on the specified observation error model so that the results are more clearly interprettable than ordinary residuals for non-normal models. DHARMA draws random predicted values from the fitted model to generate an empirical predictive density for each data point and then calculates the fraction of the empirical density that is greater than the true data point. Values of 0.5 are expected, and values near 0 or 1 indicate a mismatch between the data and the model. Particularly for the binomial and negative binomial models, in which the ordinary residuals are not normally distributed, the DHARMa residuals are a better representation of whether the data are consistent with the assumed distribution. Both the regular residuals and the DHARMa residuals are appropriate for lognormal and gamma models, since they model continuous data which is expected to be approximately normal when transformed by the link function. The DHARMA residuals should be uniformly distributed, as indicted by the QQUniform plot and the Kolmogorov-Smirnov test of uniformity. If the DHARMA residuals show significant over-dispersion then the model is not appropriate. A summary of the the DHARMa residual diagnostics is added to a table called residualTab, and models that fail the Kolmogorov-Smirnov test for uniformity of the DHARMa scaled residuals are excluded from cross-validation if residualTest is TRUE. Because the cpglm functions do not produce estimates of standard error, simulation is used to generate the DHARMa residuals if "Tweedie" is the model selected, using the rtweedie distribution from the tweedie library (Dunn and Smyth (2005)). See the functions in 3.BycatchFunctions.r for details.

For each model group the best model, as selected by the information criteria, is used to predict the mean and variance of the total bycatch in each year with the exception of the binomial, for which the model predicts to the total number of positive trips. For the binomial model, the best model is used to predict the mean and variance of the predicted probability of a positive observation in each logbook trip. Because these are predictions of the value in unobserved trips, variance is calculated as the standard error of the estimated probability squared plus the residual variance. These probabilities are summed across trips to calculate the total number of positive trips in a year, and the variances are also summed across trips. See the function makePredictionsVar in 3.BycatchFunctions.r for details. The number of positive trips is calculated because, for a very rare species that is never caught more than once in a trip, the number of positive trips would be a good estimate of total bycatch and many of the other models would fail to converge. For more common species, the estimates of total catch are more appropriate, so the results of the binomial model alone are not included in the cross-validation for model comparison.An abundance index based on the binomial distribution is also calculated, if requested.

For all the other model types, the makePredictionVar function calculates the total bycatch in each year, by first predicting the total catch (and its variance) in each trip and then summing over all trips in the logbook data. For all the negative binomial models, the log(effort) from the logbook trips is used in the predictions, along with the values of all the predictor variables, so that the model can predict bycatch and prediction interval variance in each trip directly. These are then summed over trips. For the TMBtweedie models, the CPUE is predicted for each trip in the logbook data and must be multiplied by effort and summed across trips to get the annual summaries. The variance is caclulated as the sum of the CPUE variance squared times effort squared. The cpglm function produces estimates of the predicted mean CPUE for each row of the logbook data. However, it does not produce standard error estimates. Therefore, the code produces estimated standard errors by simulation. Values of all the linear model coefficients are drawn from a multivariate normal distribution with means and a variance/covariance matrix taken from the model fit. These simulated coefficients are then multiplied by a design matrix generated from the logbook data to produce random draws of the predicted CPUE, and standard errors for each prediction are calculated as the standard deviations of the predictions. Otherwise, the total catch and its variance are estimated the same as for all the other observation error models.

For the delta-lognormal model type, the predicted means and variances of log(CPUE) are converted to the CPUE scale using the standard functions for converting between normal and logormal means and variances (See functions in 3.BycatchFunctions.r). Because these are predictions, the variances for both probability of positive catch and positive log(CPUE) are calculated as the standard error of the predicted value squared plus the residual variance. The total predicted CPUE is the predicted probability of a positive observation times the predicted positive CPUE, and predicted catch is the predicted CPUE times effort. The variance of the predictions of the total positive CPUE is calculated using the method of Lo, Jacobson, and Squire (1992) and the variance of total catch is effort squared times the variance of CPUE. The total catches and their variances are summed across trips to get the annual totals. For the delta-gamma model type, the total catch is calculated as the predicted probability of a positive catch times the predicted CPUE times effort. The variance of total catch is calculated as effort squared times the variance of the CPUE calculated using the method of Lo, Jacobson, and Squire (1992).

If a user requests an annual abundance index, this is also calculated from the best model in each model group. The annual abundance index is calculated by setting all variables other than year, and any variables required to be included in the index (e.g. region or fleet) to a reference level, which is the mean for numerical variables or the most common value for categorical variables.

Finally, this section combines all the bycatch estimates and index estimates into data frames, and prints out CSV files with the DHARMa residual diagnostics. These are P values for a Kolmogorov-Smirnoff test of whether the DHARMa residuals are uniformly distributed as expected, a test of over-dispersion, a test of zero-inflation (which is meaningless for the delta models, but helpful to see if the negative binomial model and tweedie models adequately model the zeros) and a test of whether there are more outliers than expected.

```r
#Make predictions, residuals, etc. for all models
for(mod in 1:length(modelTry)) {
  if(!is.null(modFits[[run]][[modelTry[mod]]])) {
    if(modelTry[mod] %in% c("Lognormal","Gamma")) {
       modfit1<-modFits[[run]][["Binomial"]]
       modfit2<-modFits[[run]][[modelTry[mod]]]
    } else {
       modfit1<-modFits[[run]][[modelTry[mod]]]
       modfit2<-NULL
    }
    if(EstimateBycatch) {
     modPredVals[[run]][[modelTry[mod]]]<-makePredictionsVar(modfit1=modfit1,modfit2=modfit2,modType=mo
    }
    if(EstimateIndex) {
     modIndexVals[[run]][[modelTry[mod]]]<-makeIndexVar(modfit1=modfit1,modfit2=modfit2,modType=modelT
    }
    modelTable[[run]]$formula[mod]<-
      paste(formula(modFits[[run]][[modelTry[mod]]]))[[3]]
    temp<-ResidualsFunc(modFits[[run]][[modelTry[mod]]],modelTry[mod],paste0(outVal,"Residuals",modelT
    if(!is.null(temp)) {
      residualTab[[run]][,modelTry[mod]]<-temp
      if(residualTab[[run]]["KS.p",modelTry[mod]]<0.01 & ResidualTest) modelFail[run,modelTry[mod]]<-"
    }
    if(is.null(modPredVals[[run]][[modelTry[mod]]])) modelFail[run,modelTry[mod]]<-"cv"
  } else {
    if(modelFail[run,modelTry[mod]]=="-") modelFail[run,modelTry[mod]]<-"fit"
  }
}
#Combine all predictions, except Binomial
 if(EstimateBycatch) {
  yearsumgraph<-yearSum[[run]] %>% dplyr::select(Year=Year,Total=CatEst,Total.se=Catse) %>%
    mutate(TotalVar=Total.se^2,Total.cv=Total.se/Total,TotalFixed=Total)
  allmods[[run]]<-bind_rows(c(modPredVals[[run]],list(Ratio=yearsumgraph)),.id="Source") %>%
    filter(!Source=="Binomial")
  allmods[[run]]$Valid<-ifelse(modelFail[run,match(allmods[[run]]$Source,dimnames(modelFail)[[2]])]=="
 }
 if(EstimateIndex) {
  allindex[[run]]<-bind_rows(modIndexVals[[run]],.id="Source") %>%
    filter(!Source=="Binomial")
```

```
   allindex[[run]]$Valid<-ifelse(modelFail[run,match(allindex[[run]]$Source,dimnames(modelFail)[[2]])]==
  }
 #Print the diagnostic tables
  write.csv(residualTab[[run]],paste0(outVal,"residualDiagnostics.csv"))
  write.csv(modelFail,paste0(outDir,"/modelFail.csv"))
```

The next part runs the cross validation if requested. Only observation error models that converged and produced reasonable results with the complete data set are used in cross-validation. For example, if there were not enough positive observations in all years to estimate delta-lognormal and delta-gamma models, then they will not be included in the cross-validation.

For cross-validation, the observer data are randomly divided into 10 folds. Each fold is left out one at a time and the models are fit to the other 9 folds. The same procedure described above is used to find the best model within each observation error group using information criteria and the MuMIn library if DredgeCrossValidation is TRUE, otherwise, the same variables will be used in each fold as for the full data set to save time. The fitted model is used to predict the CPUE for the left out fold, and the root mean square error is calculated as:

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where n is the number of observed trips and y is the CPUE data in the left-out tenth of the observer data, and $\hat{y}$ is the CPUE predicted from the model fitted to the other 9/10th of the observer data. The model with the lowest mean RMSE across the 10 folds is selected as the best model. Mean error is also calculated as an indicator of whether the model has any systematic bias.

$$ME = \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i - y_i$$

The last section of the main loop saves all the results to a file, and uses markdown to make output files that print all the graphics and tables using the file 6.PrintResults.rmd.

```
######## Cross validation 10 fold  ####################################
 rmsetab[[run]]<-data.frame(matrix(NA,10,length(modelTry),dimnames=list(1:10,modelTry)))
 rmsetab[[run]]<-rmsetab[[run]][,colnames(rmsetab[[run]])!="Binomial"]
 metab[[run]]<-rmsetab[[run]]
 if(DoCrossValidation & length(which(modelFail[run,colnames(modelFail)!="Binomial"]=="-"))>0) {  #Don't
  if(NumCores>3 & useParallel)  {
    cl<-makeCluster(NumCores-2)
    registerDoParallel(cl)
  }
  datval$cvsample<-sample(rep(1:10,length=dim(datval)[1]),replace=FALSE)
  table(datval$cvsample,datval$Year)
  foreach(i=1:10 ) %do% {
   datin<-datval[datval$cvsample!=i,]
   datout<-datval[datval$cvsample==i,]
   datout$SampleUnits<-rep(1,dim(datout)[1])
   for(mod in which(!modelTry %in% c("Lognormal","Gamma"))) {
     if(modelFail[run,modelTry[mod]]=="-") {
       if(DredgeCrossValidation) modfit1<-findBestModelFunc(datin,modelTry[mod])[[1]] else
        modfit1<-FitModelFuncCV(formula(paste0("y~",modelTable[[run]]$formula[mod])),modType=modelTry[m
       if(modelTry[mod]!="Binomial") {
        predcpue<-makePredictions(modfit1,modType=modelTry[mod], newdat = datout)
        rmsetab[[run]][i,modelTry[mod]]<-getRMSE(predcpue$est.cpue,datout$cpue)
        metab[[run]][i,modelTry[mod]]<-getME(predcpue$est.cpue,datout$cpue)
       } else {
        bin1<-modfit1
       }
     }
   }
   if("Lognormal" %in% modelTry | "Gamma" %in% modelTry) {
     posdat<-filter(datin,pres==1)
     for(mod in which(modelTry %in% c("Lognormal","Gamma"))) {
       if(modelFail[run,modelTry[mod]]=="-" & min(summary(posdat$Year))>0) {
         if(DredgeCrossValidation) modfit1<-findBestModelFunc(posdat,modelTry[mod])[[1]] else
          modfit1<-FitModelFuncCV(formula(paste0("y~",modelTable[[run]]$formula[mod])),modType=modelTry
         predcpue<-makePredictions(bin1,modfit1,modelTry[mod],datout)
         rmsetab[[run]][i,modelTry[mod]]<-getRMSE(predcpue$est.cpue,datout$cpue)
         metab[[run]][i,modelTry[mod]]<-getME(predcpue$est.cpue,datout$cpue)
       }
     }
   }
  }
 if(NumCores>3) stopCluster(cl)
# Calculate RMSE and ME
 modelTable[[run]]$RMSE[modelTable[[run]]$model!="Binomial"]<-apply(rmsetab[[run]],2,mean,na.rm=TRUE)
 modelTable[[run]]$ME[modelTable[[run]]$model!="Binomial"]<-apply(metab[[run]],2,mean,na.rm=TRUE)
 write.csv(residualTab[[run]],paste0(outVal,"modelSummary.csv"))
 write.csv(rmsetab[[run]],paste0(outVal,"rmse.csv"))
 write.csv(metab[[run]],paste0(outVal,"me.csv"))
 #Select best model based on cross validation
   best<-which(!is.na( modelTable[[run]]$RMSE) &
     modelTable[[run]]$RMSE==min(modelTable[[run]]$RMSE,na.rm=TRUE))
  if(length(best) >0 ) {
   bestmod[run]<-modelTry[best]
   predbestmod[[run]]<-modPredVals[[run]][[modelTry[best]]]
   indexbestmod[[run]]<-modIndexVals[[run]][[modelTry[best]]]
 } else {
   bestmod[run]<-"None"
 }
 }
 #Saving commented out for user guide.
 # save(list=c("numSp","yearSum","runName", "common", "sp","bestmod",
 #   "predbestmod","indexbestmod","allmods","allindex","modelTable",
 #  "modelSelectTable","modFits","modPredVals"
 #  ,"modIndexVals","modelFail","rmsetab","metab",
 #  "residualTab" ,"run","modelTry","EstimateIndex","EstimateBycatch",
 #   "DoCrossValidation","indexVarNames","selectCriteria","sampleUnit",
 #    "modelTry","catchType","catchUnit","residualTab",
 #    "plotValidation","trueVals","trueCols"),
 #   file=paste0(outVal,"/","resultsR"))
# rmarkdown::render("6.PrintResults.rmd",
#    params=list(outVal=outVal))
# file.rename(paste0(getwd(),"/6.PrintResults.pdf"),paste0(outVal,"/",common[run],catchType[run],"resul
# print(paste(run, common[run],"complete, ",Sys.time()))
#}
```

```
###################### End of analysis loop ######################

#Save R workspace
#if(saveR) save.image(file=paste0(outDir,"/R.workspace.rData"))
Sys.time()
```

```
## [1] "2021-07-09 15:31:49 EDT"
```

```
Sys.time()-StartTime
```

```
## Time difference of 1.21558 mins
```

Final results markdown

The R markdown file prints a .pdf file with the figures and tables to the output directory for each species labeled with the species and disposition code (e.g. SimulatedSpeciesDeadDiscardResults.pdf). These results may be all that is needed. However, if you want to look more closely at a specific model result, whether or not it was selected by the information criteria and cross-validation, all the outputs are printed to .csv files in the folders listed for each species.

```
#load(paste0(outVal,"/","resultsR"))
theme_set(theme_bw())
fignum<-1
tablenum<-1
```

Summary of results for Simulated data example for Simulated species (Genus species)

2021-07-09

```
#Print data summary
cat("Table ", tablenum,". Input data summary.", sep="")
```

Table 1. Input data summary.

```
tablenum<-tablenum+1
yearSum[[run]] %>%
  mutate_if(is.numeric,   ~ ifelse(abs(.x) > 1, round(.x), round(.x, 2))) %>%
   kbl(format="latex") %>%
   kable_styling(
     latex_options = c("hold_position","scale_down","basic")) %>%
    print()
```

```
#Print modelTable including ModelFail
cat("Table ",tablenum,". Formula of ",selectCriteria," best model, along whether models were fit succes
```

Table 2. Formula of BIC best model, along whether models were fit successfully. A dash (-) means the model converged. Failure to converge may be from data (not all years had a positive observation for delta models), fit (models did not converge) or CV (bycatch estimates had very large CVs). If cross-validation was done, mean RMSE and mean ME across folds is shown (near zero is better)

	Year	ObsCat	ObsEff	Obstrips	CPUE	CPUEse	Outlr	Pos	PosFrac	Effort	trips	EffObsFrac	tripsObsFrac	CatEst	Catse
2009	2009	6	814	33	0.01	0.01	0	3	0.09	17749	778	0.05	0.04	131	84
2010	2010	6	1374	53	0.00	0.00	0	5	0.09	13906	541	0.10	0.10	61	47
2011	2011	133	2516	81	0.09	0.03	0	22	0.27	20128	872	0.12	0.09	1064	42
2012	2012	33	568	19	0.08	0.04	0	6	0.32	17564	1034	0.03	0.02	1021	84
2013	2013	215	2327	82	0.09	0.02	0	31	0.38	20844	810	0.11	0.10	1926	53
2014	2014	6	870	27	0.01	0.00	0	5	0.19	23539	799	0.04	0.03	162	83
2015	2015	2	774	26	0.01	0.00	0	2	0.08	26741	729	0.03	0.04	69	108
2016	2016	31	1957	55	0.02	0.00	0	13	0.24	28397	805	0.07	0.07	450	59
2017	2017	2	527	14	0.00	0.00	0	1	0.07	27410	740	0.02	0.02	104	107
2018	2018	3	159	4	0.01	0.01	0	1	0.25	24204	686	0.01	0.01	455	162

```r
tablenum<-tablenum+1
df1<-modelTable[[run]] %>%
  mutate(Failure=modelFail[run,]) %>%
   mutate_if(is.numeric,round,2)
if(!DoCrossValidation) df1<-dplyr::select(df1,-c("RMSE","ME"))
kbl(df1,format="latex") %>%
   kable_styling(
     latex_options = c("hold_position","basic")) %>%
    print()
```

model	formula	RMSE	ME	Failure
Binomial	1 + Year	NA	NA	-
Lognormal	1 + Year	0.17	0	-
TMBnbinom2	1 + Year + offset(log(Effort))	0.16	0	-
TMBtweedie	1 + Year	0.16	0	-

```r
#Table residual summary
cat("Table ",tablenum,". DHARMa residual tests. Significant P values may indicate poor model specificati
```

Table 3. DHARMa residual tests. Significant P values may indicate poor model specification.

```r
tablenum<-tablenum+1
data.frame(residualTab[[run]]) %>%
 kbl(format="latex",digits = 2) %>%
   kable_styling(
     latex_options = c("hold_position","scale_down","basic")) %>%
    print()
```

	Binomial	Lognormal	TMBnbinom2	TMBtweedie
KS.D	0.03	0.09	0.03	0.05
KS.p	0.78	0.46	0.76	0.24
Dispersion.ratio	1.01	0.90	0.40	1.61
Dispersion.p	0.89	0.47	0.18	0.10
ZeroInf.ratio	1.00	NaN	1.00	1.00
ZeroInf.p	0.97	1.00	0.90	1.00
Outlier	0.00	0.00	2.00	4.00
Outlier.p	1.00	1.00	1.00	0.56

```
# Figures of All models together
if(EstimateBycatch ) {
 if(plotValidation)  plotFitsValidate(filter(allmods[[run]],Valid==1),trueVals,NULL,trueCols[run]) else
    plotFits(filter(allmods[[run]],Valid==1),"All",NULL)
  cat("\n Figure ",fignum,". Total bycatch estimates for all valid models, including a simple unstratifi
  fignum<-fignum+1
}
```

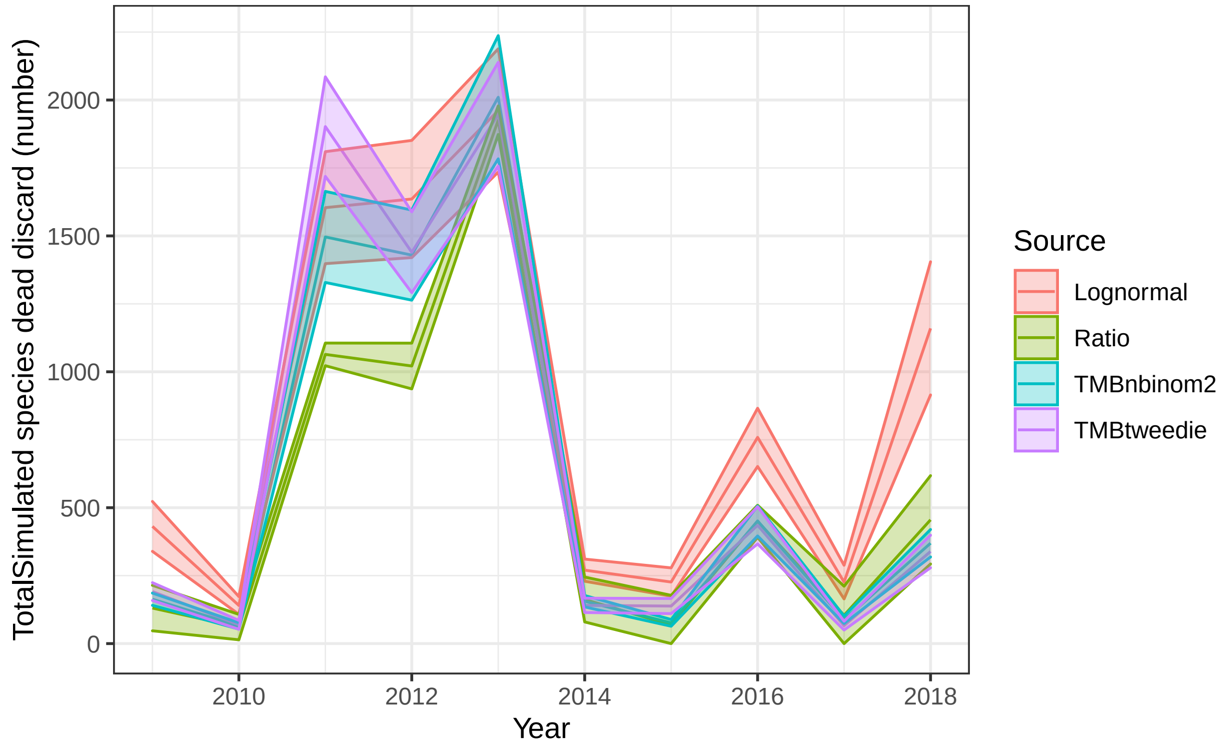

Figure 1. Total bycatch estimates for all valid models, including a simple unstratified ratio estimator, for Simulated species.

```
if(EstimateIndex & any(allindex[[run]]$Valid==1)) {
 plotIndex(filter(allindex[[run]],Valid==1),"All",NULL)
 cat("\n Figure ",fignum,". Abundance indices from all valid models for ",common[run],". \n",sep="")
 fignum<-fignum+1
}
```

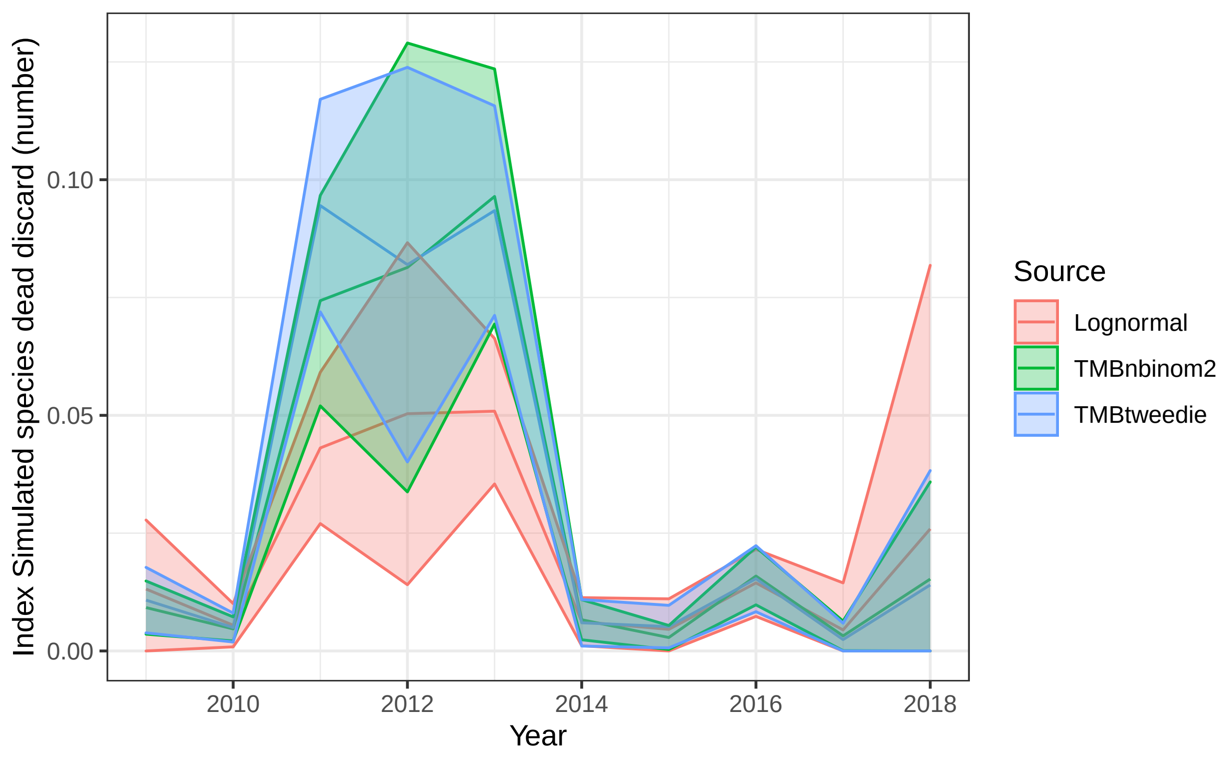

Figure 2. Abundance indices from all valid models for Simulated species.

```
#Show cross validation figures
if(DoCrossValidation &!all(is.na(modelTable[[run]]$RMSE))) {
 plotCrossVal(rmsetab[[run]],metab[[run]],NULL)
 cat("\n Figure ",fignum,". Cross validation results for ",common[run],". Lowest RMSE model is ",bestmod
 fignum<-fignum+1
}
```

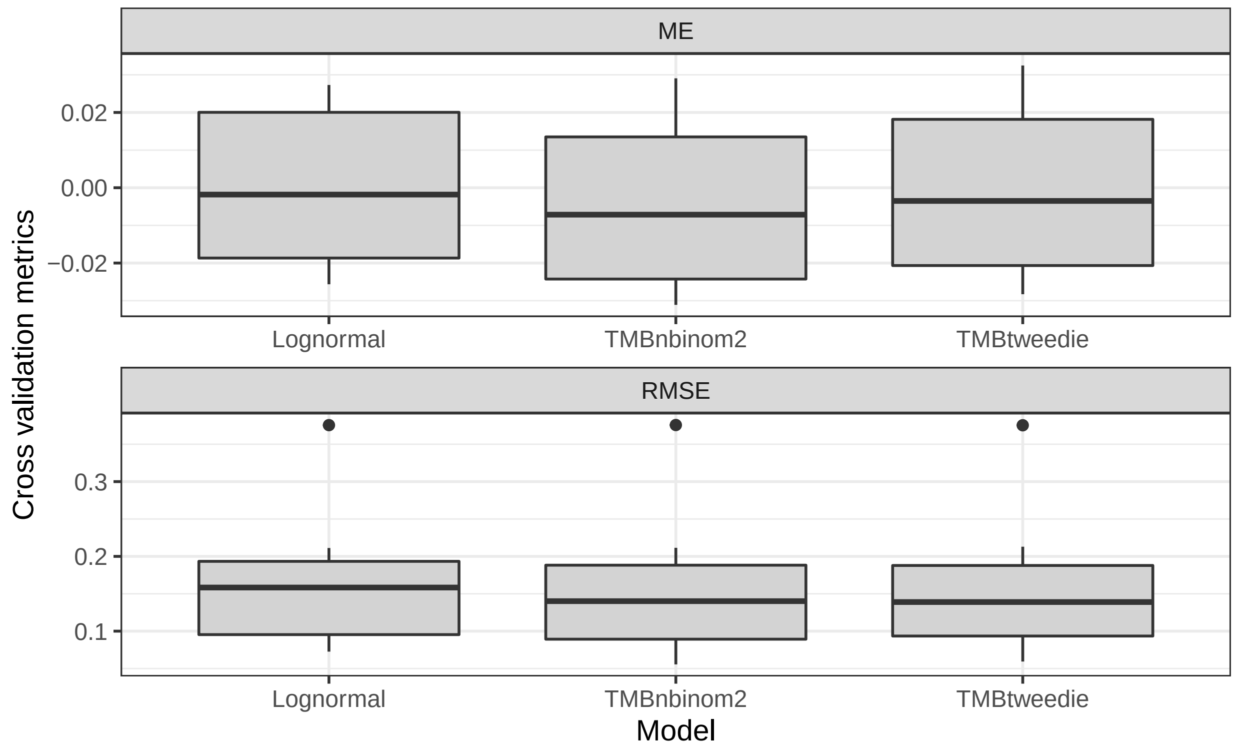

Figure 3. Cross validation results for Simulated species. Lowest RMSE model is TMBnbinom2.

```
#Print for each model type
for(mod in 1:length(modelTry)) {
if(modelFail[run,mod]=="-") {
  cat("\n Table ",tablenum,". Model selection table for ",modelTry[mod],". Weights are calculated based
  tablenum<-tablenum+1
  data.frame(modelSelectTable[[run]][[modelTry[mod]]]) %>%
   mutate_if(is.numeric,~ ifelse(abs(.x) > 1, round(.x,1), round(.x, 2))) %>%
   kbl(format="latex") %>%
   kable_styling(
     latex_options = c("hold_position","scale_down","basic")) %>%
    print()
   temp<-ResidualsFunc(modFits[[run]][[modelTry[mod]]],modelTry[mod],fileName=NULL)
   cat("\n Figure ",fignum,". Residuals for the ",selectCriteria, " best model for ",        modelTry[mo
   fignum<-fignum+1
  if(EstimateBycatch) {
   if(plotValidation & modelTry[mod]!="Binomial")  plotFitsValidate(filter(allmods[[run]],Source==model
   plotFits(modPredVals[[run]][[modelTry[mod]]],modelTry[mod],fileName=NULL)
  if(modelTry[mod]=="Binomial")  cat("\n Figure ",fignum,". Estimated total number of positive ",sampleU
   cat("\n Figure ",fignum,". Estimated total bycatch from ",modelTry[mod]," plus and minus one standard
   fignum<-fignum+1
 }
 if(EstimateIndex) {
   plotIndex(modIndexVals[[run]][[modelTry[mod]]],modelTry[mod],fileName=NULL)
   cat("\n Figure ",fignum,". Estimated relative index from ",modelTry[mod],". \n",sep="")
   fignum<-fignum+1
 }
}
}
```

Table 4. Model selection table for Binomial. Weights are calculated based on BIC.

	X.Intercept.	season	Year	season.Year	AICc	AIC	BIC	df	logLik	selectCriteria	delta	weight
0	-2.3	NA	+	NA	412.8	412.3	452	10	-196.1	452	0.0	0.95
1	-2.3	+	+	NA	414.9	414.2	458	11	-196.1	458	5.9	0.05
3	-2.3	+	+	+	420.7	418.4	498	20	-189.2	498	46.0	0.00

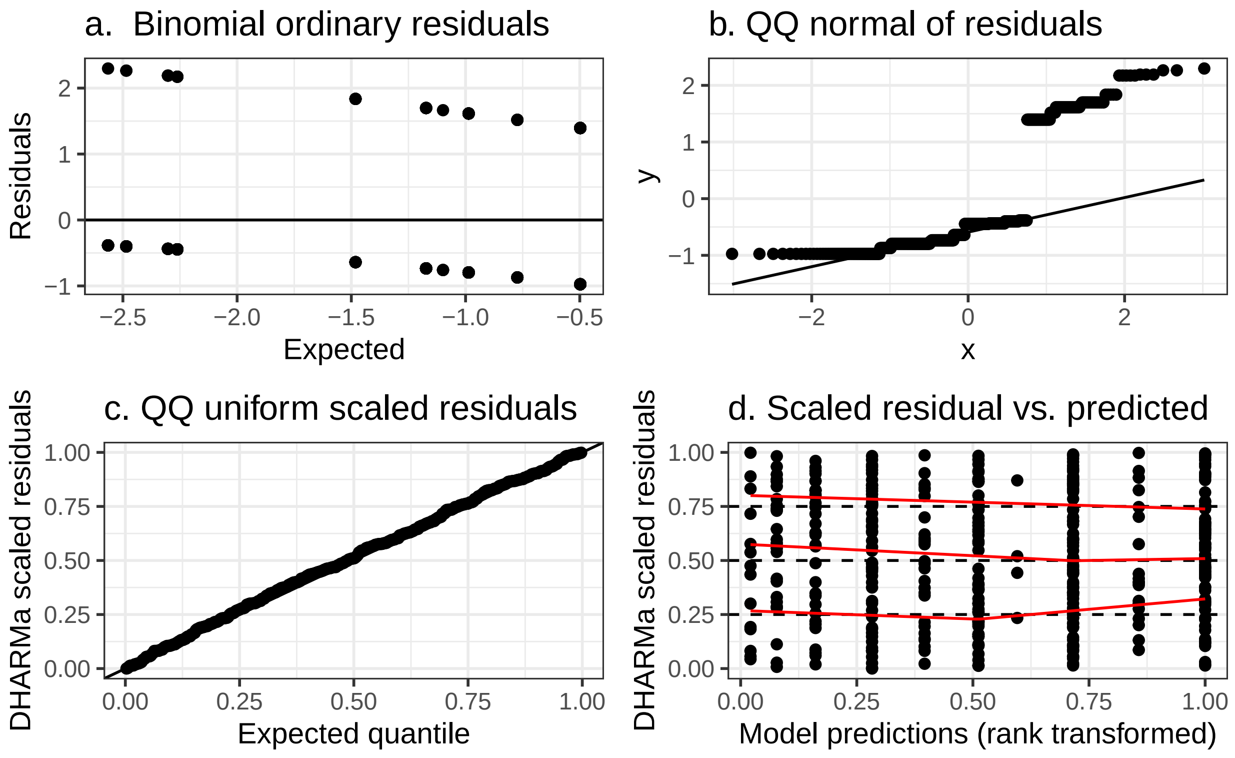

Figure 4. Residuals for the BIC best model for Binomial.

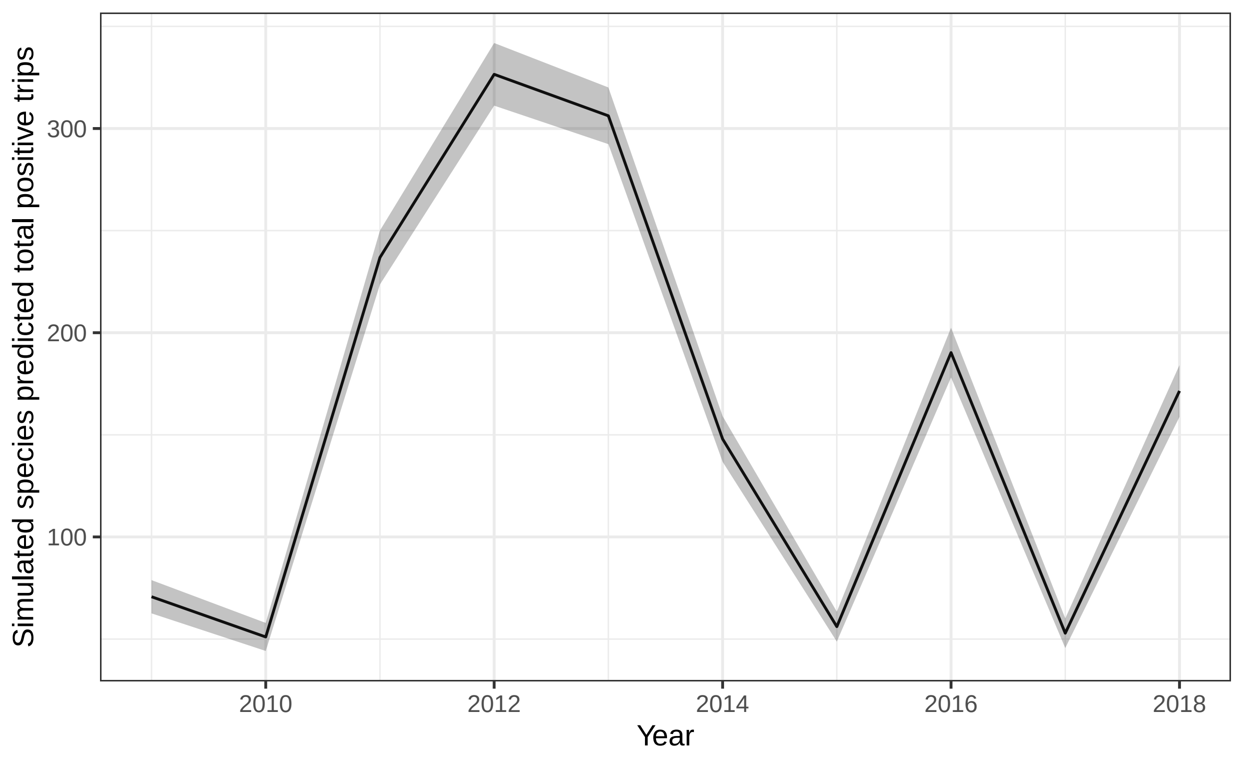

Figure 5. Estimated total number of positive trips from Binomial plus and minus one standard error.

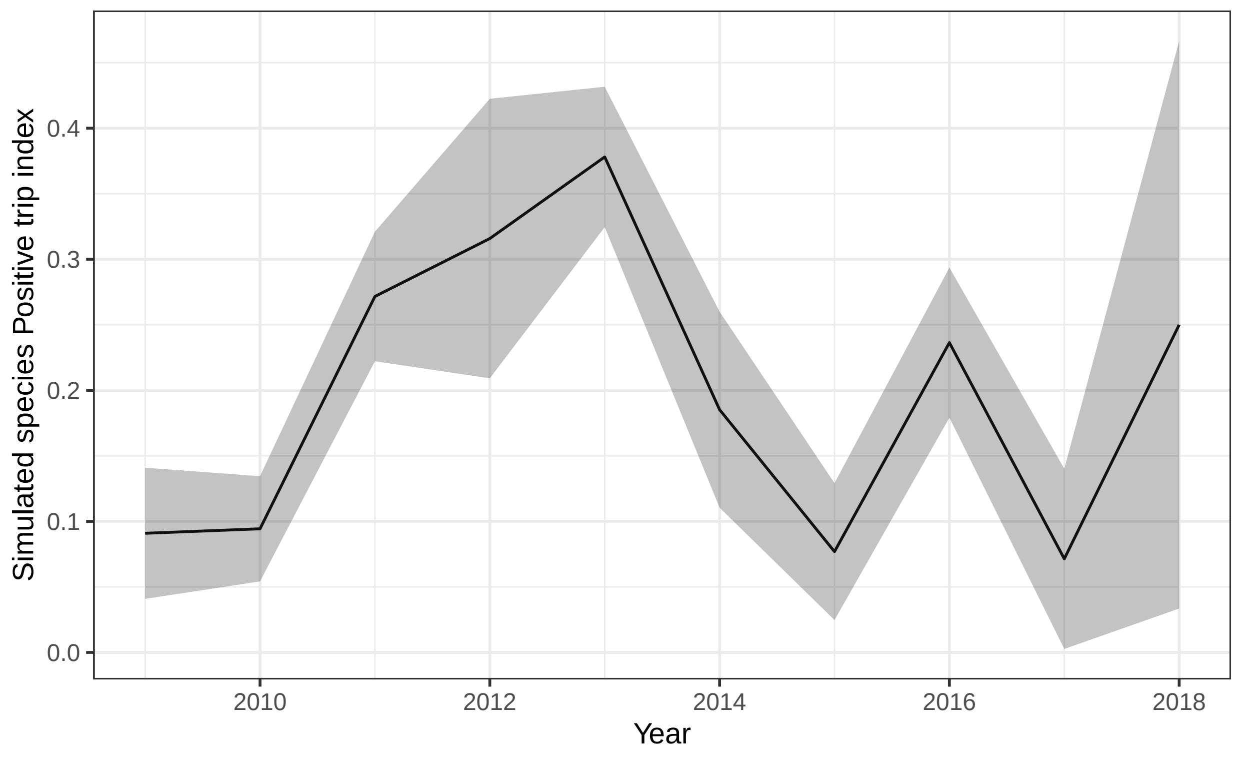

Figure 6. Estimated relative index from Binomial.

Table 5. Model selection table for Lognormal. Weights are calculated based on BIC.

	X.Intercept.	season	Year	AICc	AIC	BIC	df	logLik	selectCriteria	delta	weight
0	-2.1	NA	+	285.8	282.4	309.8	11	-130.2	309.8	0.0	0.69
1	-2.1	+	+	285.6	281.5	311.4	12	-128.8	311.4	1.6	0.31

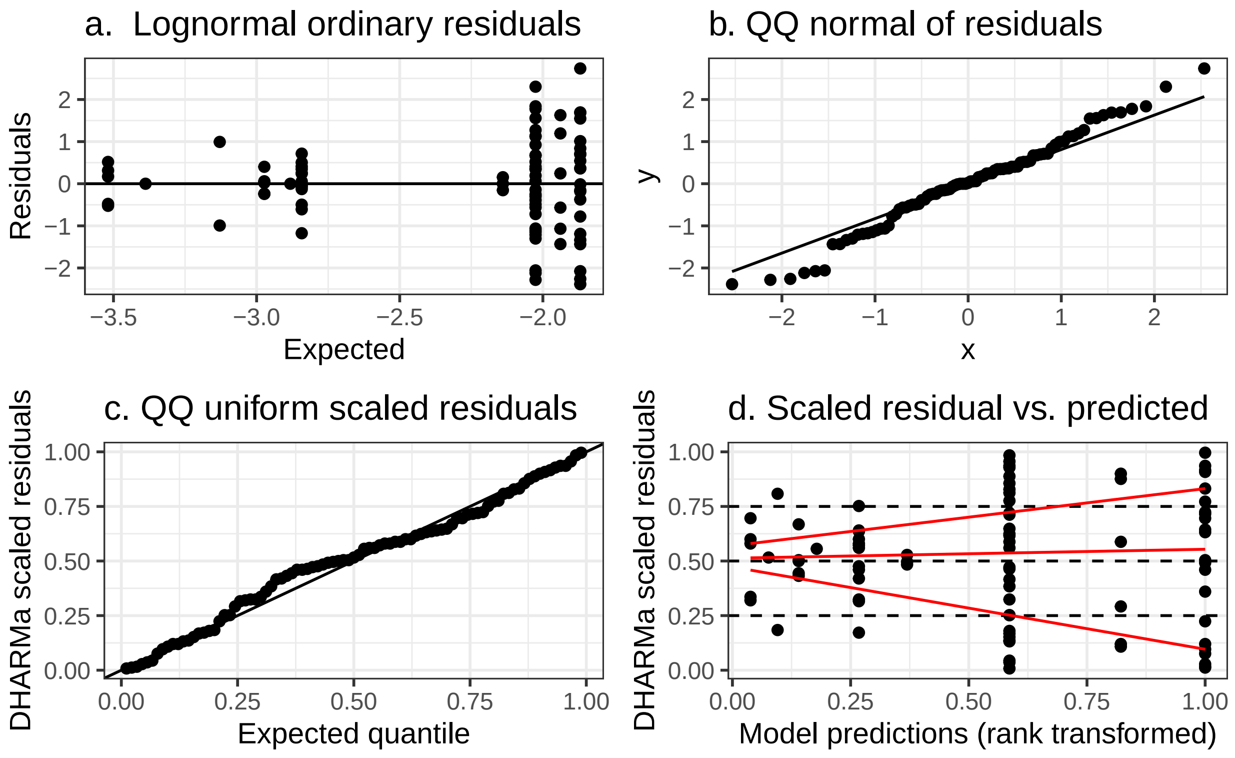

Figure 7. Residuals for the BIC best model for Lognormal.

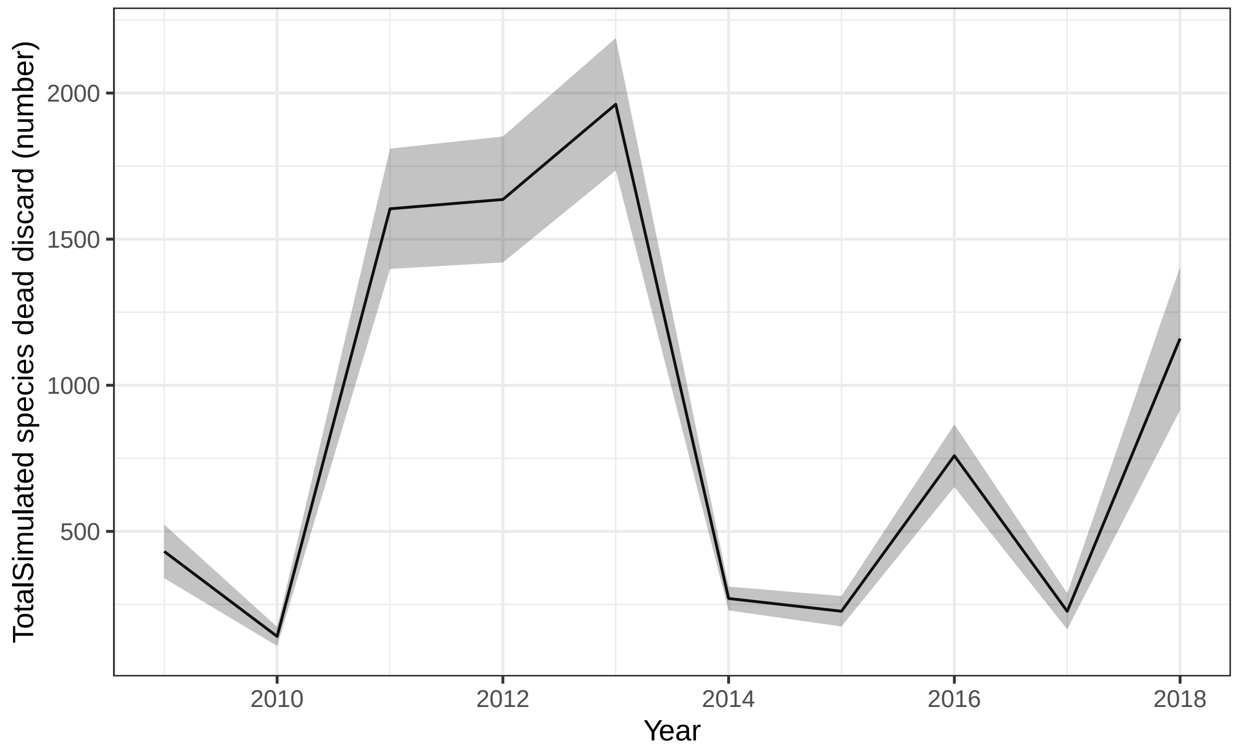

Figure 8. Estimated total bycatch from Lognormal plus and minus one standard error.

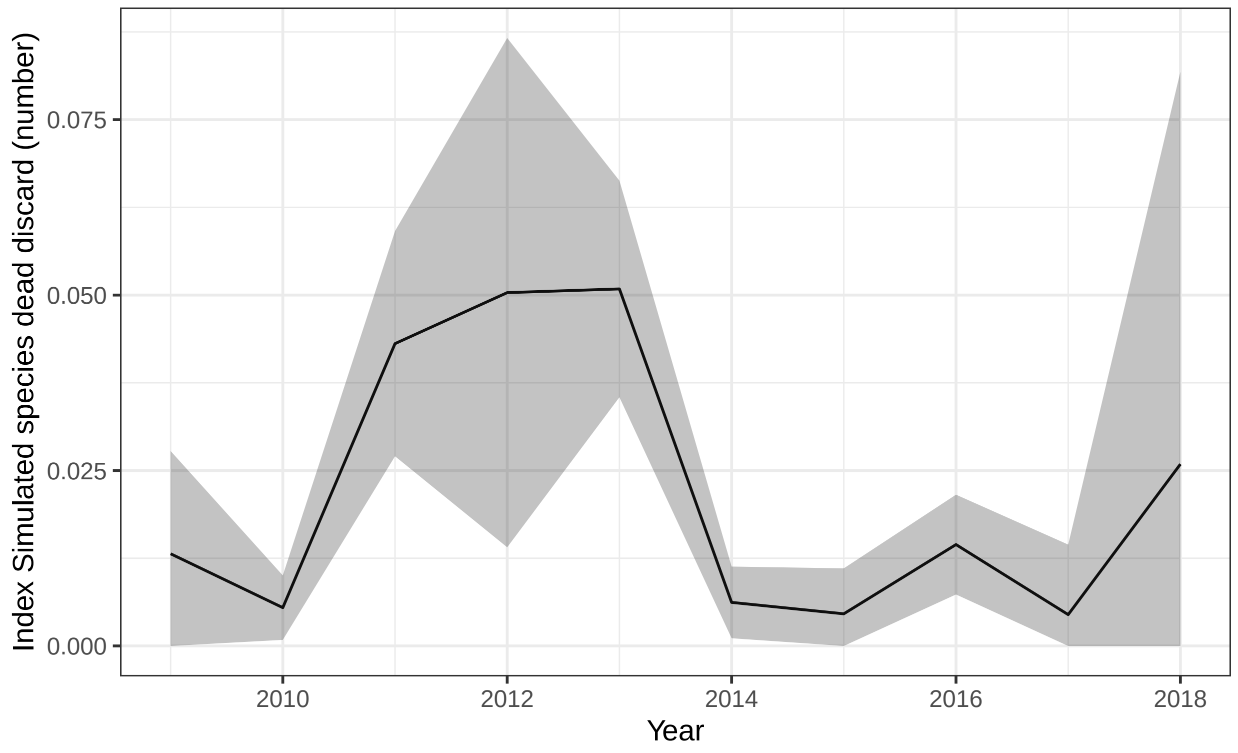

Figure 9. Estimated relative index from Lognormal.

Table 6. Model selection table for TMBnbinom2. Weights are calculated based on BIC.

	cond..Int..	disp..Int..	cond.season.	cond.Year.	cond.season.Year.	cond.offset.log.Effort...	AICc	AIC	BIC	df	logLik	selectCriteria	delta	weight
0	-4.7	+	NA	+	NA	+	815.6	814.9	858.6	11	-396.4	858.6	NA	NA
1	-4.7	+	+	+	NA	+	816.2	815.4	863.1	12	-395.7	863.1	NA	NA
3	-4.6	+	+	+	+	+	NA	NA	NA	21	NA	NA	NA	NA

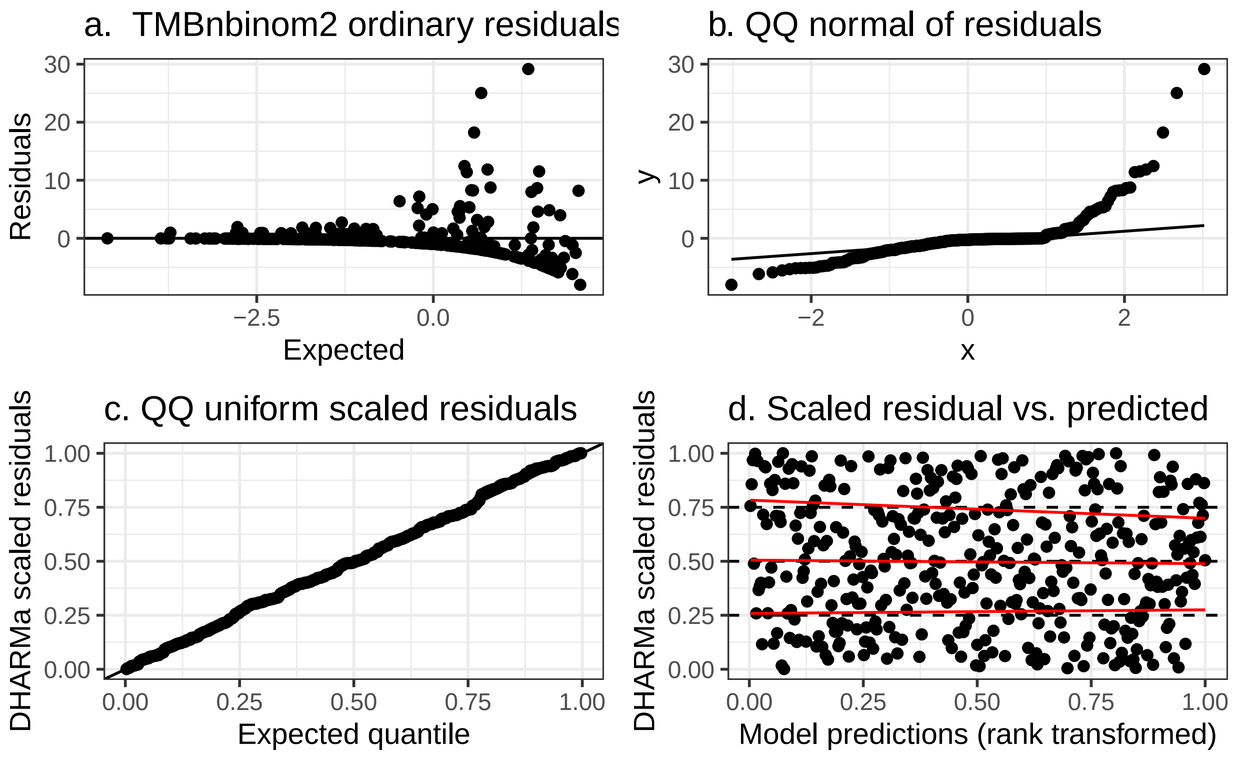

Figure 10. Residuals for the BIC best model for TMBnbinom2.

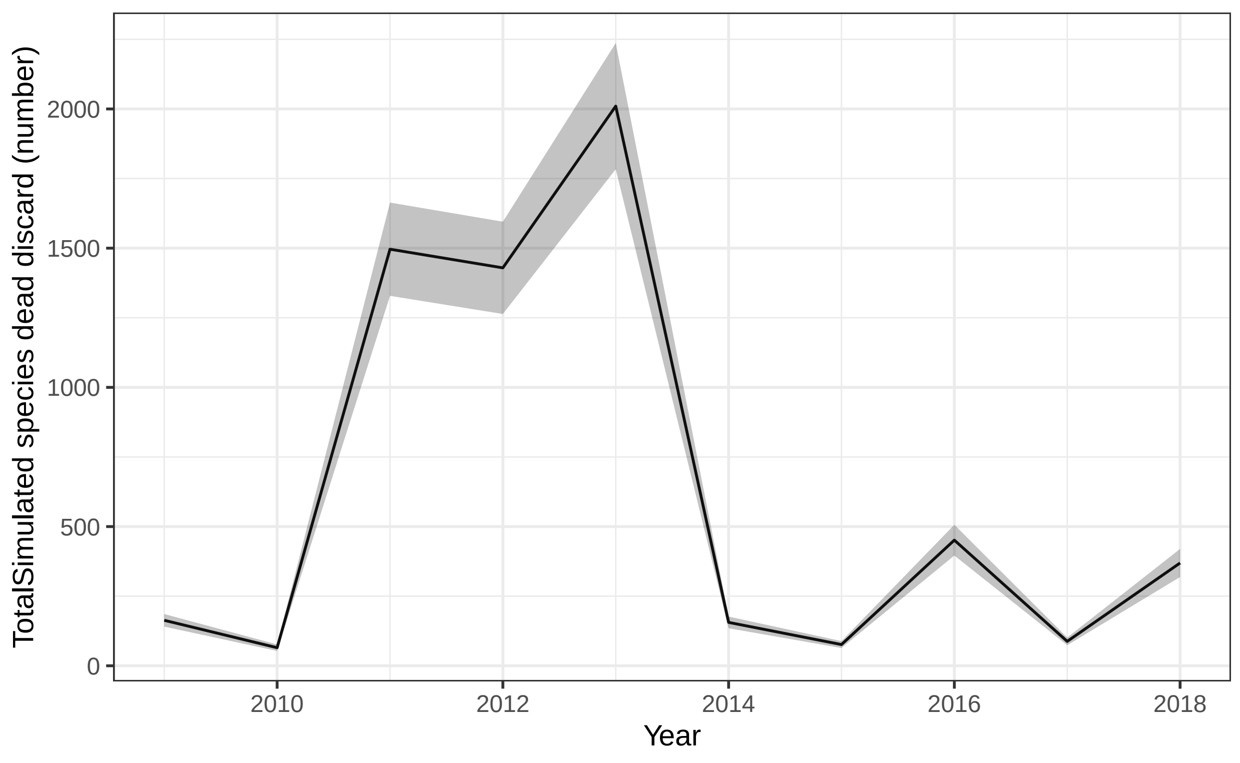

Figure 11. Estimated total bycatch from TMBnbinom2 plus and minus one standard error.

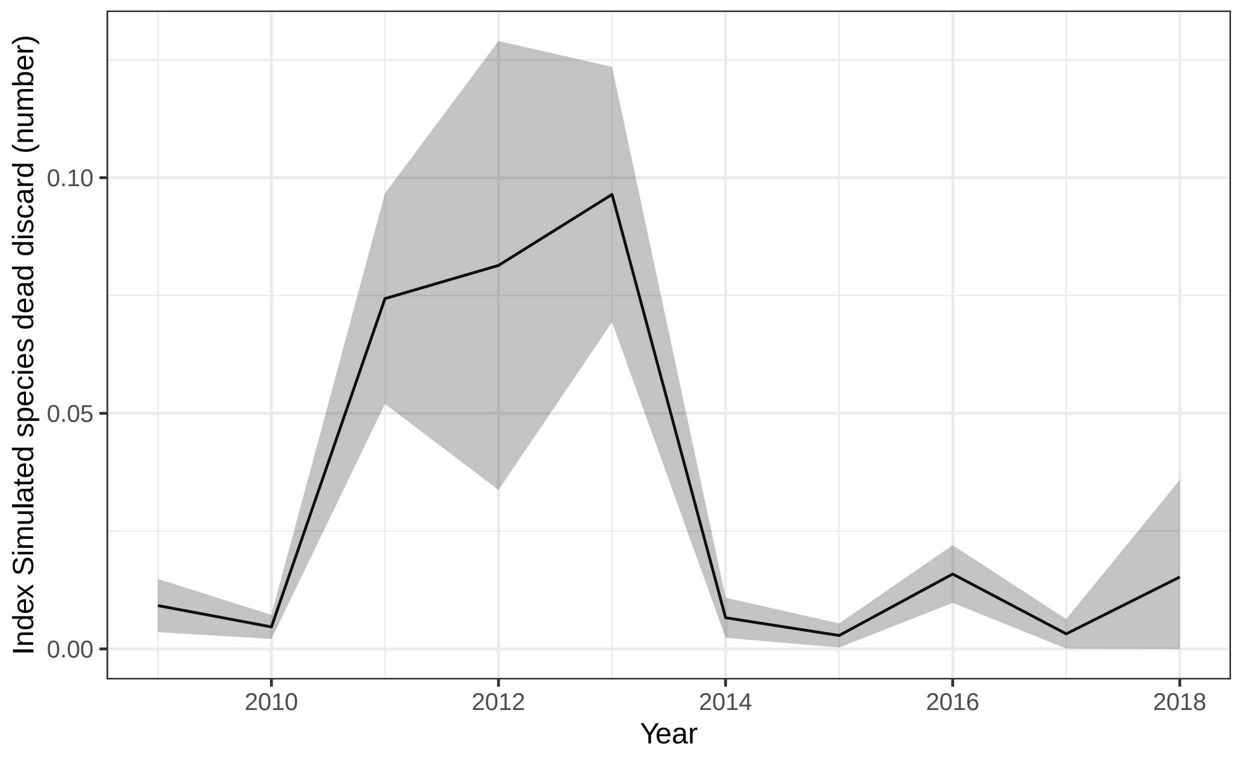

Figure 12. Estimated relative index from TMBnbinom2.

Table 7. Model selection table for TMBtweedie. Weights are calculated based on BIC.

	cond..Int..	disp..Int..	cond.season.	cond.Year.	cond.season.Year.	AICc	AIC	BIC	df	logLik	selectCriteria	delta	weight
0	-4.5	+	NA	+	NA	288.7	287.9	335.6	12	-131.9	335.6	NA	NA
1	-4.5	+	+	+	NA	286.3	285.4	337.1	13	-129.7	337.1	NA	NA
3	-4.5	+	+	+	+	NA	NA	NA	22	NA	NA	NA	NA

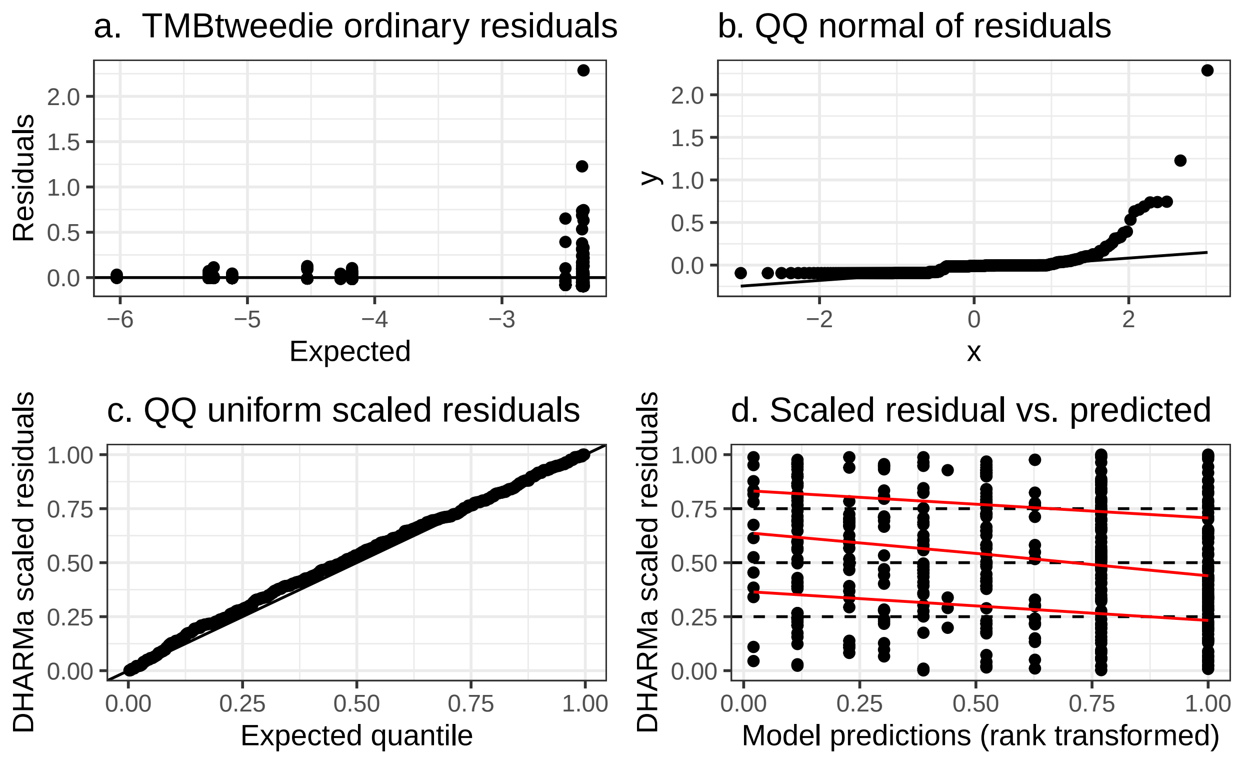

Figure 13. Residuals for the BIC best model for TMBtweedie.

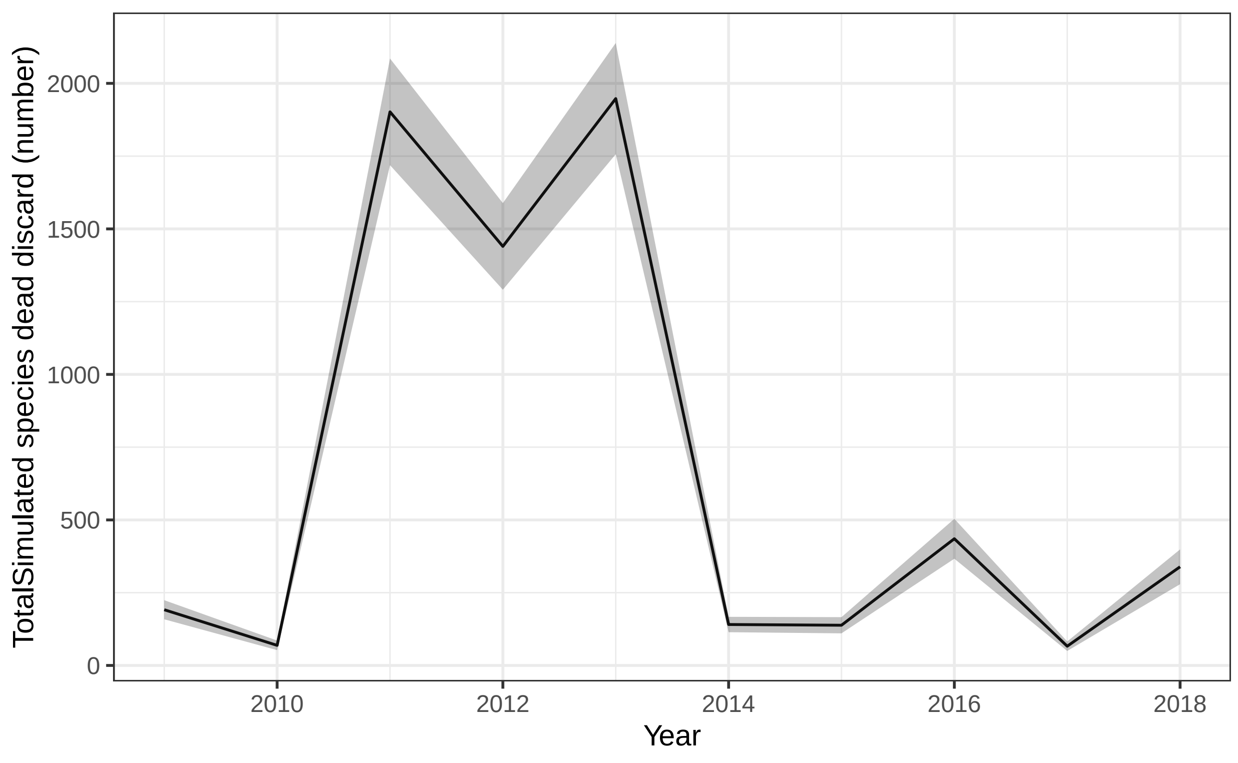

Figure 14. Estimated total bycatch from TMBtweedie plus and minus one standard error.

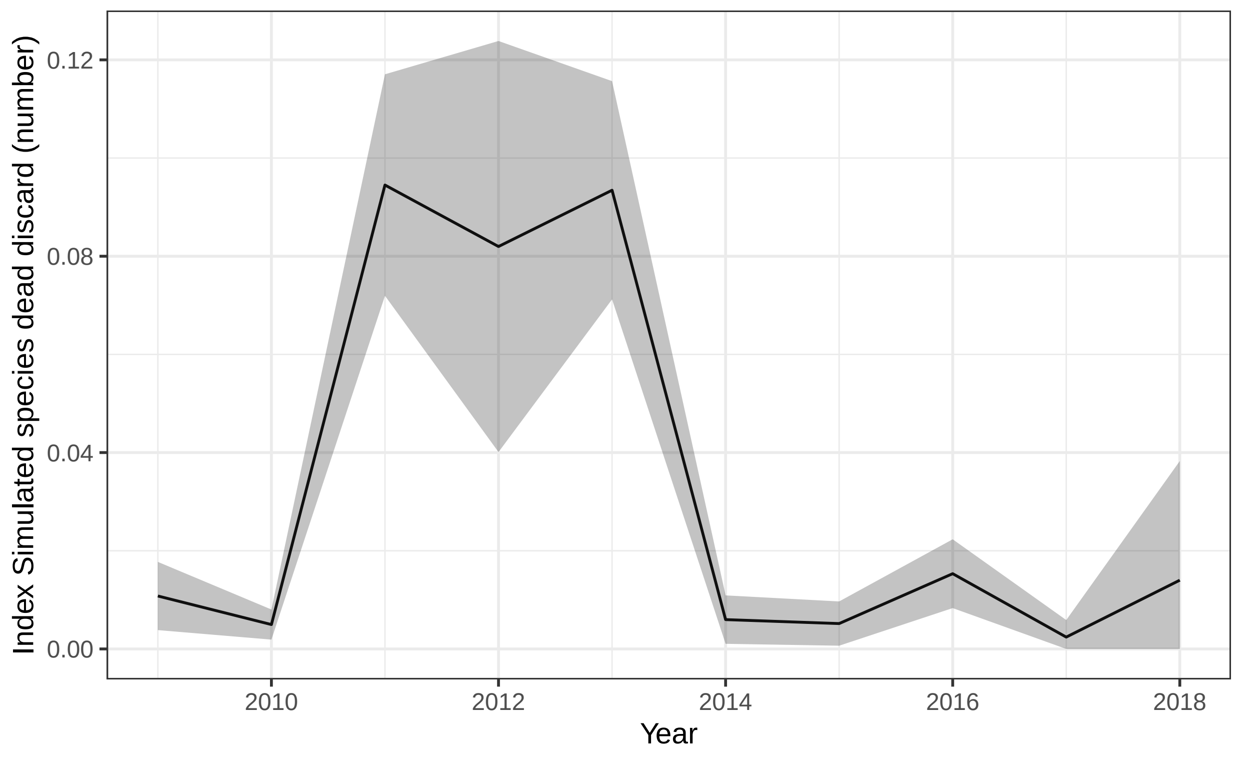

Figure 15. Estimated relative index from TMBtweedie.

Conclusions

Although this code automates the whole process of model selection, it is not recommended that the final model be used without looking closely at the outputs. The selected model must have good fits and should be reasonably consistent with data according to the DHARMa residuals. The results should appear reasonable and be around the same scale as the ratio estimator results. The model should not be overly complex. Also, look at the model selection table to see if other models are also supported by the data.

References

Auguie, Baptiste. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics.* https://CRAN.R-project.org/package=gridExtra.

Barton, Kamil. 2020. *MuMIn: Multi-Model Inference.* https://CRAN.R-project.org/package=MuMIn.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Brooks, Mollie E., Kasper Kristensen, Koen J. van Benthem, Arni Magnusson, Casper W. Berg, Anders Nielsen, Hans J. Skaug, Martin Maechler, and Benjamin M. Bolker. 2017. "glmmTMB Balances Speed and Flexibility Among Packages for Zero-Inflated Generalized Linear Mixed Modeling." *The R Journal* 9 (2): 378–400. https://journal.r-project.org/archive/2017/RJ-2017-066/index.html.

Corporation, Microsoft, and Steve Weston. 2020. *doParallel: Foreach Parallel Adaptor for the 'Parallel' Package.* https://CRAN.R-project.org/package=doParallel.

Dunn, Peter K., and Gordon K. Smyth. 2005. "Series Evaluation of Tweedie Exponential Dispersion Models." *Statistics and Computing* 15: 267–80.

Hartig, Florian. 2020. *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models.* https://CRAN.R-project.org/package=DHARMa.

Henry, Lionel, and Hadley Wickham. 2020. *Tidyselect: Select from a Set of Strings.* https://CRAN.R-project.org/package=tidyselect.

Iannone, Richard, Joe Cheng, and Barret Schloerke. 2020. *Gt: Easily Create Presentation-Ready Display Tables.* https://CRAN.R-project.org/package=gt.

Lo, N. C. H., L. D. Jacobson, and J. L. Squire. 1992. "Indexes of Relative Abundance from Fish Spotter Data Based on Delta-Lognormal Models." Journal Article. *Canadian Journal of Fisheries and Aquatic Sciences* 49 (12): 2515–26. https://doi.org/Doi%2010.1139/F92-278.

Microsoft, and Steve Weston. 2020. *Foreach: Provides Foreach Looping Construct.* https://CRAN.R-project.org/package=foreach.

Ooms, Jeroen. 2020. *Pdftools: Text Extraction, Rendering and Converting of PDF Documents.* https://CRAN.R-project.org/package=pdftools.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

RStudio Team. 2020. *RStudio: Integrated Development Environment for r.* Boston, MA: RStudio, PBC. http://www.rstudio.com/.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with s.* Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wickham, Hadley. 2007. "Reshaping Data with the reshape Package." *Journal of Statistical Software* 21 (12): 1–20. http://www.jstatsoft.org/v21/i12/.

———. 2016. *Ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, and Thomas Lin Pedersen. 2019. *Gtable: Arrange 'Grobs' in Tables.* https://CRAN.R-project.org/package=gtable.

Xie, Yihui. 2015. *Dynamic Documents with r and Knitr.* Second. Chapman; Hall/CRC.

———. 2019. "TinyTeX: A Lightweight, Cross-Platform, and Easy-to-Maintain LaTeX Distribution Based on TeX Live." *TUGboat*, no. 1: 30–32. http://tug.org/TUGboat/Contents/contents40-1.html.

———. 2021. *Tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents.* https://github.com/yihui/tinytex.

Zhang, Yanwei. 2013. "Likelihood-Based and Bayesian Methods for Tweedie Compound Poisson Linear Mixed Models."