Bycatch Estimator User Guide

Elizabeth A. Babcock

2021-05-04

ebabcock@rsmas.miami.edus (mailto: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, grid, gtable, gt, 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 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 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 disposisitons (e.g. dead discard, live release) from the same fishery simultaneously if they are all being estimated from the same datasets.

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

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 trip or 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 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 dataset, with an error message.

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 bycatc
h.
# 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). If estimating an abundance index, give a formula including the variables that should be included in the index. This can be just year, or, if separate indices are desired for multiple areas or fleets, these variables can be included.

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

#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 obser
vers
#sample 100% of effort in sampled trips.
obsEffort<-"sampled.sets"
#obsEffortNotSampled<-"unsampled.sets"  #Not needed yet. May be added to later versio
n
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 uni
ts (trips or sets)
# that each row includes. If the logbook data is not aggregated (i.e. each row is a #
sample unit) just make this NA
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 len
gth 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 co
mpare all
# intermediate models using information criteria. Use indexModel to specify which str
ata to
# keep separate in calculating abundance indices.
complexModel<-formula(y~Year+fleet+hbf+area+w.SWO+year:area)
simpleModel<-formula(y~Year+fleet+area)
indexModel<-formula(y~Year+area)

## The variables must have identical names and factor levels in the observer and logb
ook datasets
#Specify which of these variables should be interpreted as categorical to make sure t
hey are in factor format.
```

```
#Variables not in this list will retain their original format
factorNames=c("Year","season")
```

Finally, 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)). 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 dataset. This saves time with large datasets.

Also, specifiy whether to save the R workspace after the models run. 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 in the example. 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.

```
#Specify which observation error models to try. Options are delta-lognormal, delta-ga
mma, negative
#binomial and tweedie, specified as: "Lognormal" for delta lognormal,"Gamma" for delt
a gamm,"NegBin" for negative binomial
#using glm.mb in the MASS library, "Tweedie" for cpglm, and TMB nbinom1, nbinom2, and
tweedie in the glmmTMB
#library, specified with "TMB" followed by the model type. Binomial is run
#automatically as part of the delta models.
modelTry<-c("Lognormal","Gamma","NegBin","Tweedie","TMBnbinom1","TMBnbinom2","TMBtwee
die")

#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 dataset. DredgeCrossValidation specifies whether to use information cr
iteria
#to find the best model in cross validation, using the dredge function, or just keep
the same model formula.
#Do not use dredge for very large datasets, as the run will be slow.
DoCrossValidation<-TRUE
DredgeCrossValidation<-FALSE

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

# Specify whether to save R workspace. This should be true unless you
# don't have space on your disk.

saveR<-TRUE
```

Preliminary set up and data summaries

Once the specification file is set up, open the file 2.ByCatchModels. The first section loads libraries and does some book-keeping, then prints a summary of the input data. This is all run by sourcing a file called “4.PreliminaryDataSummaries.r.” If your computer has muliple cores, the code will run in parellel on up to the number of cores minus 2. If you have trouble with this, comment out the line that say: #NumCores<-detectCores() #if(NumCores>3) registerDoParallel(NumCores-2)

The data summaries are output to directories named “output” followed by the specified run name. A pdf file with summaries for all species 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 n
ot matter
# how the logbook effort data is aggregated, as long has it includes data on all stra
tification 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 th
e
# 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.BycatchModelSpe
cificationExample.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 specFil
e line by line, but it
# will also be sourced again later. The file will be saved, with the addition of the
date, to the output directory

################################################################################
### From here on no changes should be needed.  However, there are places where you sh
ould stop and look at
#  the output before proceeding.

################################################################################
#Load required libraries
library(tidyverse)
library(ggplot2)
library(MASS)
library(lme4)
library(cplm)
library(DHARMa)
library(tidyselect)
library(MuMIn)
library(gridExtra)
library(grid)
library(gtable)
library(gt)
library(pdftools)
library(foreach)
library(doParallel)
library(tweedie)
library(reshape2)
library(glmmTMB)
```

```
#########################################################################
#Read in data specification
source(specFile)
#Make data summary outputs
source(paste0(baseDir,"/4.preliminaryDataSummary.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 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.

This section may be slow if you have a large dataset 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.

```
############## This is the main analysis loop ##########################
#The following runs all the models and prints all the output files.
bestmod<-NULL
predbestmod<-list()
allmods<-list()
allindex<-list()
modelTable<-list()
modelSelectTable<-list()
modelFail<-matrix("-",numSp,length(modelTry)+1,dimnames=list(common,c("Binomial",mode
lTry)))
rmsetab<-list()
metab<-list()
outFiles<-list()
residualTab<-list()
#for(run in 1:numSp) {
run<-1  #For user guide, set run to 1 instead of looping
 datval<-dat[[run]]
 residualTab[[run]]<-matrix(0,8,length(modelTry)+1,dimnames=list(c("KS.D","KS.p", "Di
spersion.ratio","Dispersion.p" ,
                            "ZeroInf.ratio" ,"ZeroInf.p","Outlier" , "Outlier.p"),
                            c("Binomial",modelTry)))
 modelTable[[run]]<-data.frame(model=c("Binomial",modelTry),
     formula=rep("",length(modelTry)+1),
     RMSE=rep(NA,length(modelTry)+1),
   ME=rep(NA,length(modelTry)+1))
 outVal<-paste0(dirname[[run]],common[run],catchType[run])
 outFiles[[run]]<-c(fileList[[run]],
   paste0(outVal,rep(c("Total","Index","Residuals","ModelSelection"),length(modelTr
y)+1),
     rep(dimnames(modelFail)[[2]],each=3),".pdf"),
   paste0(outVal,"AllTotal.pdf"),
   paste0(outVal,"AllIndex.pdf"))
 if(DoCrossValidation) outFiles[[run]]<-c(outFiles[[run]],paste0(outVal,"Crossvalidat
ion.pdf"),paste0(outVal,"BestTotal.pdf","BestIndex.pdf"))
 varExclude<-NULL
 predvals<-rep(list(NULL),length(modelTry))
 names(predvals)<-modelTry
 indexvals<-rep(list(NULL),length(modelTry))
 names(indexvals)<-modelTry
 modfits<-rep(list(NULL),length(modelTry))
 names(modfits)<-modelTry
```

After setting up the data and reprinting the summary, if either delta-lognormal or delta-gamma is requested, the first model run is the binomial. The models are fitted using the glm function with a logit link. 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.

This best model, as selected by the information criteria, is then 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.

Finally, summary plots are printed to the output files. These include the predicted number of positive trips plus and minus the standard deviation of the prediction, residual plots, residual qqnormal plots, and residuals calculated using the DHARMa R library (Hartig (2020)). The DHARMa library uses simulation to generate scaled residuals based on the specified observation error model so that the results are more clearly interpretable. 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 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. The DHARMA residuals should be uniformly distributed. 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 of the DHARMa scaled residuals are excluded from cross-validation if residualTest is TRUE.

If all years have at least one positive observation, the loop next runs the lognormal and gamma models (if requested) and calculates total bycatch using the delta-lognormal and delta-gamma methods. For the lognormal model, the CPUE is log transformed, and the mean CPUE for positive observations is modeled with the lm() function, using the MuMin dredge() function to find the best set of predictor variables. The predicted means and variances of log(CPUE) are then 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 gamma method, the default inverse link is used to model the positive CPUE values. 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).

The estimated total catches are plotted for both the delta-lognormal and delta-gamma models, along with the residuals and DHARMa residuals for the models fitted to the positive observations. 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 next part runs all the models that are applied to all data together (i.e. not delta models). 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 cases, so that the model predicts:

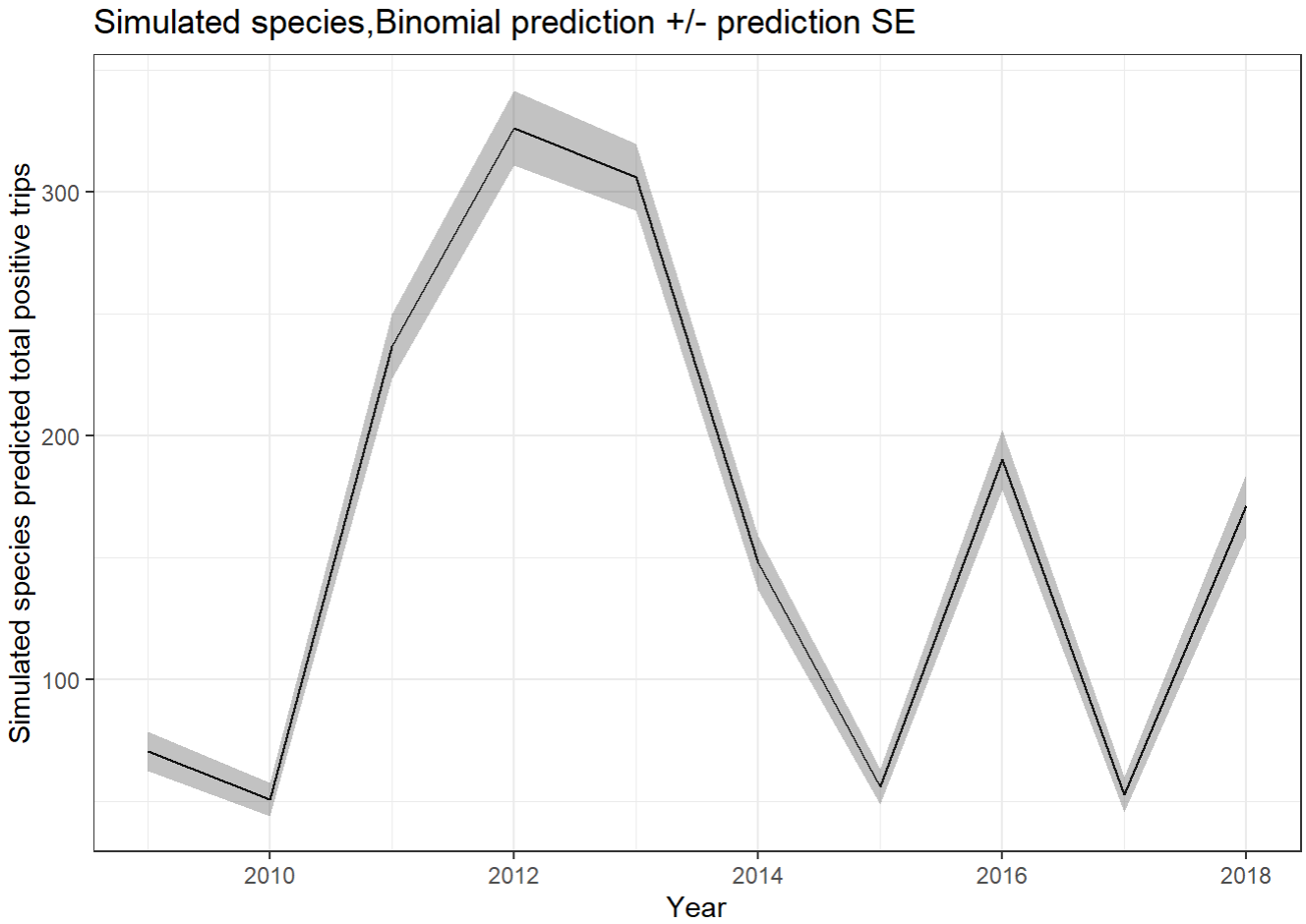
$$log(C_i) = b_0 + b_1 x_1 + offset(log(E_i))$$

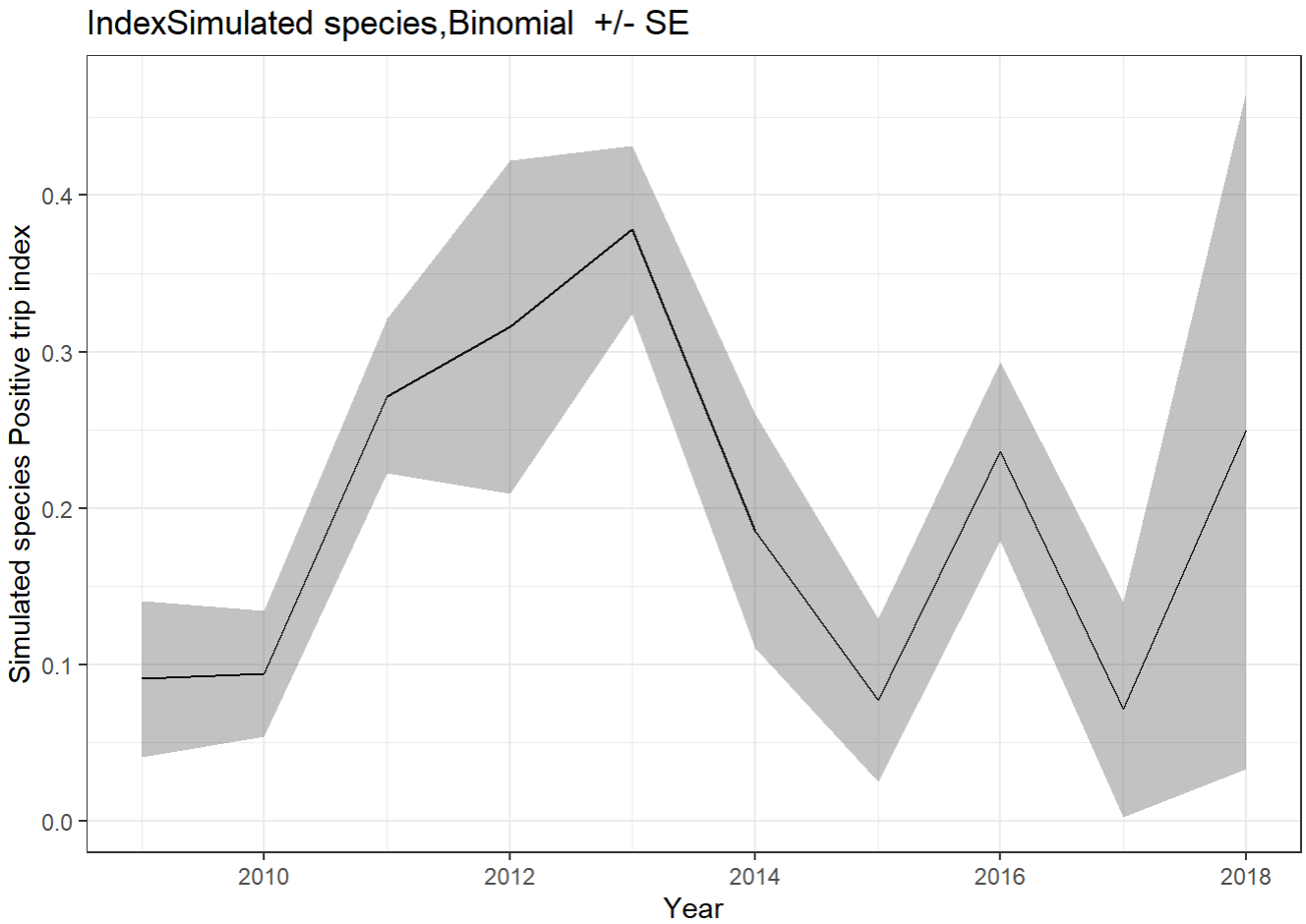
were C_i is the catch in trip i in the observer data, $b_0 + b_1 x_1$ is an example linear predictor with an intercept and a slope, and the offset is the log of the effort E_i in each trip. This is algebraically equivalent to modeling CPUE as a function of the same linear predictor. The total catches and variances are predicted for each trip in the logbook data, using the log of the trip's effort as an offset, and are summed across trips to get the annual summaries.

The total catch estimates are plotted, along with the ordinary and DHARMa residuals. For the negative binomial model. Because the response is an integer, the ordinary residuals tend to have patterns that are difficult to interpret. Thus, the DHARMa residuals are more useful for evaluating model fit for negative binomial models.

The Tweedie distribution can also be used to model the predicted CPUE, using the cpglm function in the cplm library (Zhang (2013)) or the tweedie family in glmmTMB (Brooks et al. (2017)). The libraries give very similar results. 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. 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. Simulation is also used to generate the DHARMa residuals, using the rtweedie distribution from the tweedie library (Dunn and Smyth (2005)). See the functions in 3.BycatchFunctions.r for details.

```
#Fit delta models
 if("Lognormal" %in% modelTry | "Gamma" %in% modelTry) {  #Delta models if requested
  bin1<-findBestModelFunc(datval,"Binomial",printOutput=TRUE)
  if(!is.null(bin1)) {  #If model converged make predictions and/or index and print f
igures
   bintab<-bin1[[2]]
   bin1<-bin1[[1]]
   if(EstimateBycatch) {
    binpredvals<-makePredictionsVar(bin1,modType="Binomial",newdat=logdat,printOutput
=TRUE)
    plotFits(binpredvals,"Binomial",paste0(outVal,"TotalBinomial.pdf"))
   }
   if(EstimateIndex) {
    binindexvals<-makeIndexVar(bin1,modType="Binomial",printOutput=TRUE)
    plotIndex(binindexvals,"Binomial",paste0(outVal,"IndexBinomial.pdf"))
   }
   residualTab[[run]][,"Binomial"]<-ResidualsFunc(bin1,"Binomial",paste0(outVal,"Resi
dualsBinomial.pdf"))
   if(residualTab[[run]]["KS.p","Binomial"]<0.01 & ResidualTest) modelFail[run,"Binom
ial"]<-"resid"
   if(is.null(binpredvals)) modelFail[run,"Binomial"]<-"cv"
   modelTable[[run]]$formula[1]<-paste(formula(bin1))[[3]]
   } else  {
     modelFail[run,"Binomial"]<-"fit"
     binpredvals<-NULL
     binindexvals<-NULL
   }
   posdat<-filter(dat[[run]],pres==1)
   y<-unlist(lapply(posdat[,factorNames],function(x) length(setdiff(levels(x),x)))) #
See if all levels are included
   varExclude<-names(y)[y>0]
   if(length(varExclude>0)) print(paste(common[run], "excluding variable",varExclud
e,"from delta models for positive catch"))
   if(min(summary(posdat$Year))>0 &!is.null(bin1)) { #If all years have at least one
positive observation and binomial converged, carry on with delta models
     foreach(mod=which(modelTry %in% c("Lognormal","Gamma"))) %do% {
       modfits[[modelTry[mod]]]<-findBestModelFunc(posdat,modelTry[mod],printOutput=T
RUE)
     }
   } else {
     print("Not all years have positive observations, skipping delta models")
     modelFail[run,c("Lognormal","Gamma")]<-"data"
     modelFail[run,"Binomial"]<-"na"
     binpredvals<-NULL
     binindexvals<-NULL
   }
 }
```


Simulated species,Binomial prediction +/- prediction SE
Simulated species predicted total positive trips
300
200
100
2010
2012
2014
2016
2018
Year


IndexSimulated species,Binomial +/- SE
Simulated species Positive trip index
0.4
0.3
0.2
0.1
0.0
2010
2012
2014
2016
2018
Year

```
## [[1]]
## [[1]][[1]]
##
## Call:
## lm(formula = y ~ 1 + Year, data = obsdatval, na.action = na.fail)
##
## Coefficients:
## (Intercept)     Year2010     Year2011     Year2012     Year2013     Year2014
##     -2.1396      -0.8334       0.2703       0.2010       0.1141      -1.3793
##    Year2015     Year2016     Year2017     Year2018
##     -0.9891      -0.7027      -1.2484      -0.7425
##
##
## [[1]][[2]]
## Global model call: lm(formula = formula(modfit1), data = obsdatval, na.action = n
a.fail)
## ---
## Model selection table
##   (Intrc) seasn Year  AICc   AIC   BIC df   logLik selectCriteria delta weight
## 1   -2.14          + 285.8 282.4 309.8 11 -130.202          309.8  0.00  0.693
## 2   -2.14     +    + 285.6 281.5 311.4 12 -128.772          311.4  1.63  0.307
## Models ranked by selectCriteria(x)
##
##
## [[2]]
## [[2]][[1]]
##
## Call:  glm(formula = y ~ season + 1 + Year, family = "Gamma", data = obsdatval,
##     na.action = na.fail)
##
## Coefficients:
## (Intercept)      season2     Year2010     Year2011     Year2012     Year2013
##       8.430        3.296        8.723       -5.773       -4.887       -5.947
##    Year2014     Year2015     Year2016     Year2017     Year2018
##      21.326        4.993        5.639       17.881        6.127
##
## Degrees of Freedom: 88 Total (i.e. Null);  78 Residual
## Null Deviance:       130.3
## Residual Deviance: 80.36     AIC: -127.2
##
## [[2]][[2]]
## Global model call: glm(formula = formula(modfit1), family = "Gamma", data = obsdat
val,
##     na.action = na.fail)
## ---
## Model selection table
##   (Intrc) seasn Year   AICc    AIC    BIC df logLik selectCriteria delta weight
## 2    8.43     +    + -123.1 -127.2 -97.37 12 75.618          -97.4  0.00  0.889
## 1    8.43          + -117.1 -120.6 -93.20 11 71.288          -93.2  4.17  0.111
## Models ranked by selectCriteria(x)
```

```
#Fit all models except delta
foreach(mod = which(!modelTry %in% c("Lognormal","Gamma"))) %do% {
  modfits[[modelTry[mod]]]<-findBestModelFunc(datval,modelTry[mod],printOutput=TRUE)
}
```

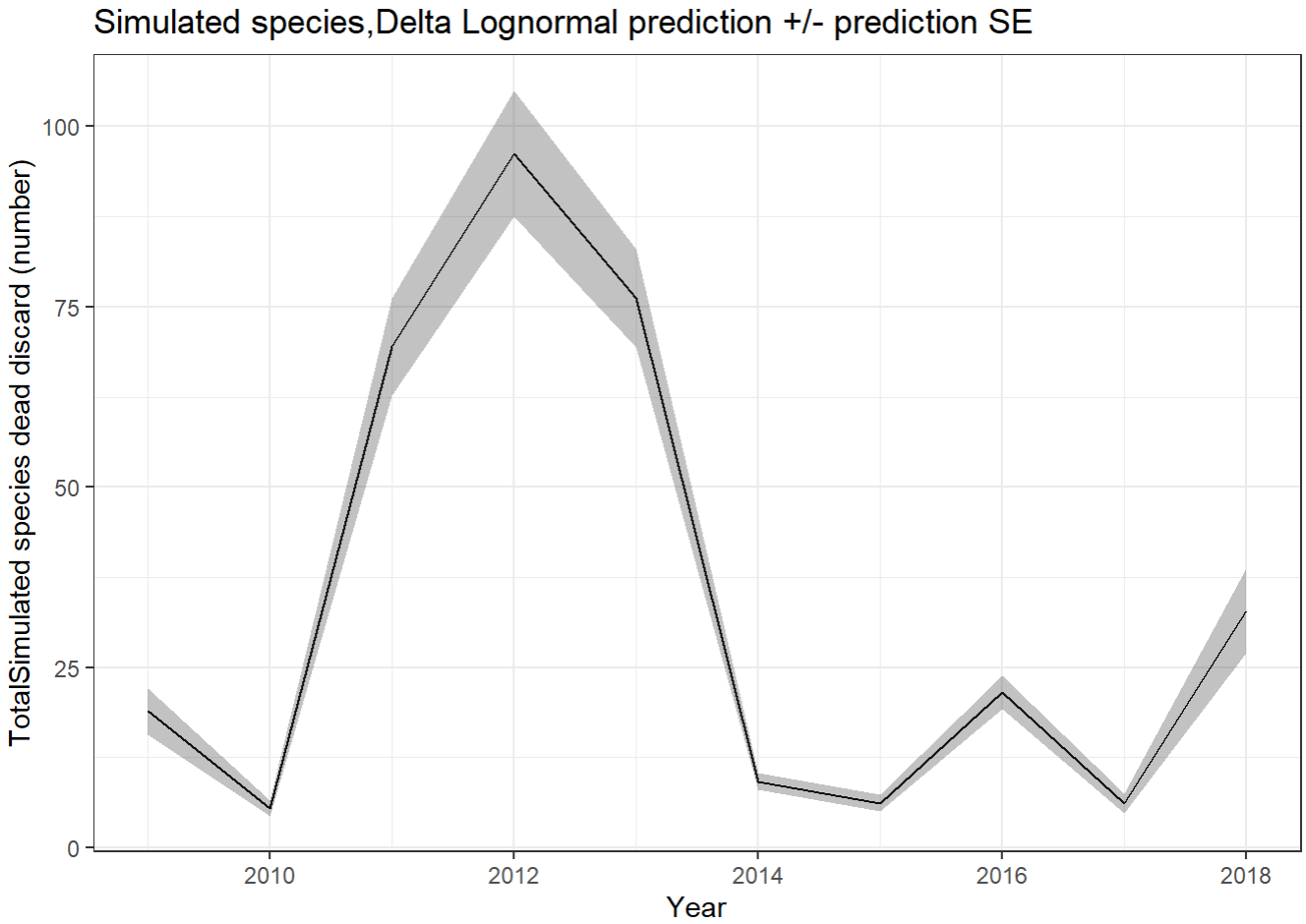
```
## [[1]]
## [[1]][[1]]
##
## Call:  glm.nb(formula = y ~ 1 + Year + offset(log(Effort)), data = obsdatval,
##     na.action = na.fail, control = glm.control(epsilon = 1e-06,
##         maxit = 30), init.theta = 0.1698656324, link = log)
##
## Coefficients:
## (Intercept)     Year2010     Year2011     Year2012     Year2013     Year2014
##     -4.6881      -0.6774       2.0889       2.1795       2.3493      -0.3290
##    Year2015     Year2016     Year2017     Year2018
##     -1.1714       0.5462      -1.0567       0.5056
##
## Degrees of Freedom: 393 Total (i.e. Null);  384 Residual
## Null Deviance:       274.5
## Residual Deviance: 207.9     AIC: 814.9
##
## [[1]][[2]]
## Global model call: glm.nb(formula = formula(modfit1), data = obsdatval, na.action
= na.fail,
##     control = glm.control(epsilon = 1e-06, maxit = 30), init.theta = 0.1838139694,
##     link = log)
## ---
## Model selection table
##    (Int) ssn Yer ssn:Yer off(log(Eff))  AICc   AIC   BIC df   logLik
## 1 -4.688       +                    + 815.6 814.9 858.6 11 -396.431
## 2 -4.684   +   +                    + 816.2 815.4 863.1 12 -395.714
## 4 -4.645   +   +       +            + 826.7 824.2 907.7 21 -391.109
##   selectCriteria delta weight
## 1          858.6  0.00  0.906
## 2          863.1  4.54  0.094
## 4          907.7 49.12  0.000
## Models ranked by selectCriteria(x)
##
##
## [[2]]
## [[2]][[1]]
##
## Call:
## cpglm(formula = y ~ 1 + Year, data = obsdatval, na.action = na.fail)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1059  -1.1027  -0.6419  -0.5285   4.4532
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.5296     0.6654  -6.807 3.85e-11 ***
## Year2010     -0.7751     0.9216  -0.841  0.40089
## Year2011      2.1705     0.7096   3.059  0.00238 **
## Year2012      2.0283     0.8492   2.389  0.01739 *
## Year2013      2.1592     0.7094   3.044  0.00250 **
## Year2014     -0.5905     1.0819  -0.546  0.58552
```

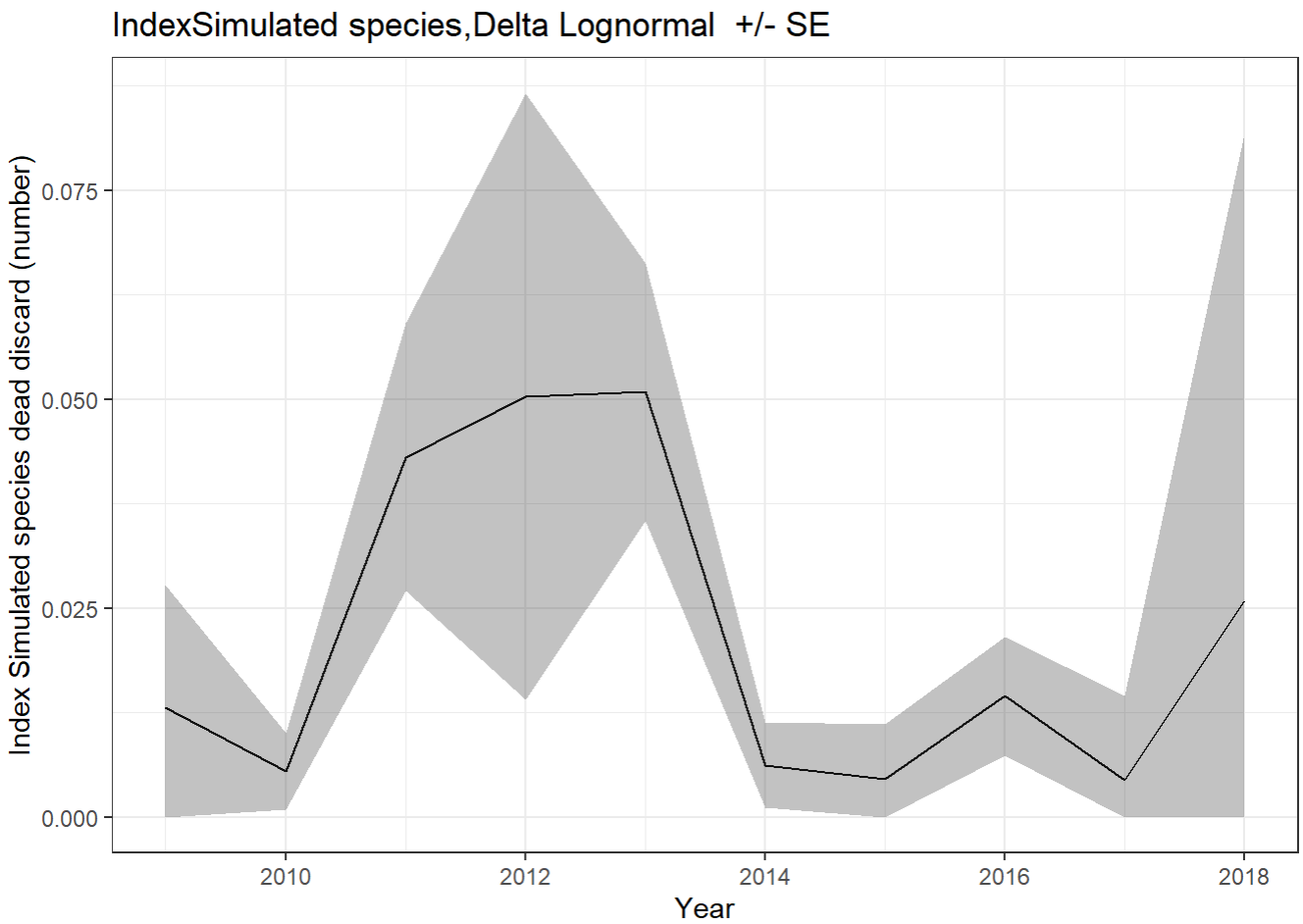
```
## Year2015     -0.7361     1.1206  -0.657  0.51166
## Year2016      0.3516     0.8158   0.431  0.66672
## Year2017     -1.4974     1.6286  -0.919  0.35844
## Year2018      0.2612     1.9099   0.137  0.89128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Estimated dispersion parameter: 1.4147
## Estimated index parameter: 1.4987
##
## Residual deviance: 323.09  on 384  degrees of freedom
## AIC:  283.87
##
## Number of Fisher Scoring iterations:  7
##
##
## [[2]][[2]]
## Global model call: cpglm(formula = formula(modfit1), data = obsdatval, na.action =
na.fail)
## ---
## Model selection table
##    (Int) ssn Yer ssn:Yer  AICc   AIC   BIC df   logLik selectCriteria delta
## 1 -4.530       +         284.4 283.9 323.6 10 -131.937          323.6  0.00
## 2 -4.521   +   +         282.1 281.4 325.1 11 -129.685          325.1  1.47
## 4 -4.499   +   +       + 292.7 290.5 370.0 20 -125.238          370.0 46.36
##   weight
## 1  0.676
## 2  0.324
## 4  0.000
## Models ranked by selectCriteria(x)
##
##
## [[3]]
## [[3]][[1]]
## Formula:          y ~ 1 + Year + offset(log(Effort))
## Data: obsdatval
##       AIC       BIC    logLik  df.resid
##  837.1004  880.8402 -407.5502       383
##
## Number of obs: 394
##
## Overdispersion parameter for nbinom1 family (): 8.83
##
## Fixed Effects:
##
## Conditional model:
## (Intercept)     Year2010     Year2011     Year2012     Year2013     Year2014
##    -4.21947     -0.03013      1.04363      1.20003      1.66192      0.43416
##    Year2015     Year2016     Year2017     Year2018
##    -0.38679      0.66591     -0.64717      0.71235
##
```

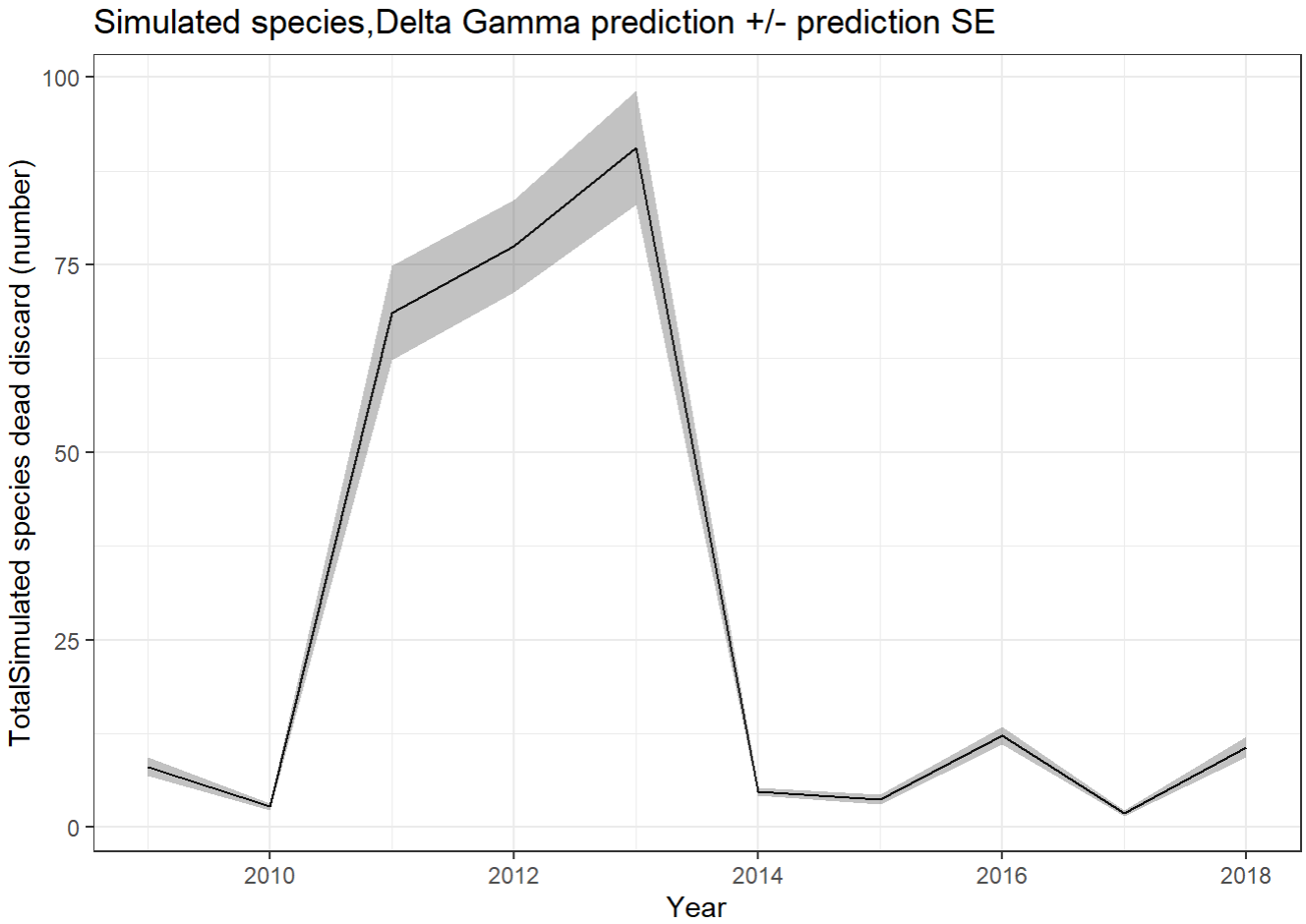
```
## [[3]][[2]]
## Global model call: glmmTMB(formula = y ~ season + Year + season:Year + offset(log
(Effort)),
##     data = obsdatval, family = TMBfamily, na.action = na.fail,
##     ziformula = ~0, dispformula = ~1)
## ---
## Model selection table
##   cnd((Int)) dsp((Int)) cnd(ssn) cnd(Yer) cnd(ssn:Yer) cnd(off(log(Eff)))  AICc
## 1     -4.219          +                +                                + 837.8
## 2     -4.219          +        +       +                                + 839.9
## 4     -4.157          +        +       +            +                   +
##     AIC   BIC df  logLik selectCriteria delta
## 1 837.1 880.8 11 -407.55          880.8  0.00
## 2 839.1 886.8 12 -407.55          886.8  5.98
## 4             21
## Models ranked by selectCriteria(x)
##
##
## [[4]]
## [[4]][[1]]
## Formula:          y ~ 1 + Year + offset(log(Effort))
## Data: obsdatval
##       AIC       BIC    logLik  df.resid
##  814.8622  858.6021 -396.4311       383
##
## Number of obs: 394
##
## Overdispersion parameter for nbinom2 family (): 0.17
##
## Fixed Effects:
##
## Conditional model:
## (Intercept)     Year2010     Year2011     Year2012     Year2013     Year2014
##     -4.6881      -0.6774       2.0890       2.1795       2.3493      -0.3290
##    Year2015     Year2016     Year2017     Year2018
##     -1.1714       0.5462      -1.0567       0.5055
##
## [[4]][[2]]
## Global model call: glmmTMB(formula = y ~ season + Year + season:Year + offset(log
(Effort)),
##     data = obsdatval, family = TMBfamily, na.action = na.fail,
##     ziformula = ~0, dispformula = ~1)
## ---
## Model selection table
##   cnd((Int)) dsp((Int)) cnd(ssn) cnd(Yer) cnd(ssn:Yer) cnd(off(log(Eff)))  AICc
## 1     -4.688          +                +                                + 815.6
## 2     -4.684          +        +       +                                + 816.2
## 4     -4.645          +        +       +            +                   +
##     AIC   BIC df   logLik selectCriteria delta
## 1 814.9 858.6 11 -396.431          858.6  0.00
## 2 815.4 863.1 12 -395.714          863.1  4.54
```

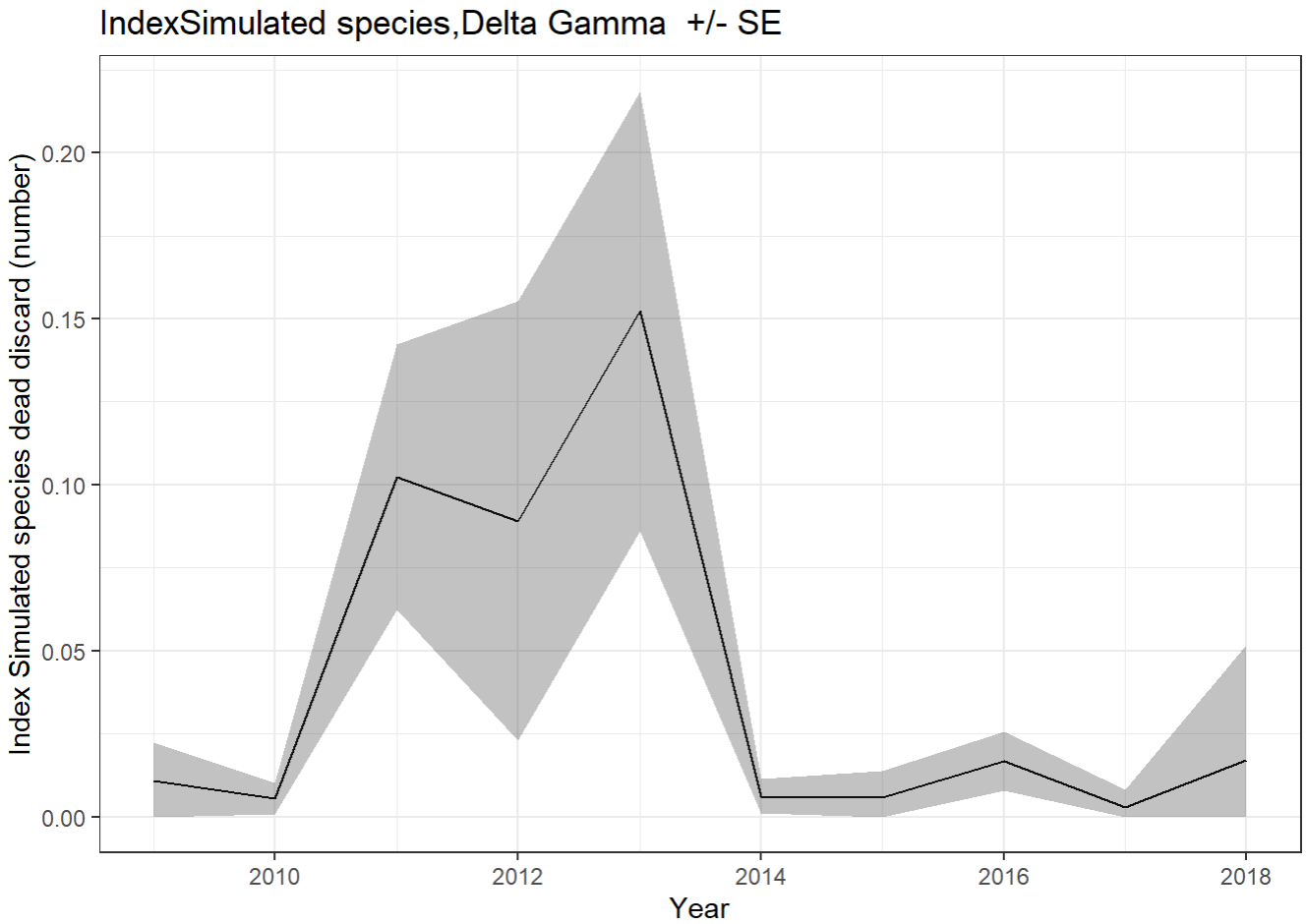
```
## 4             21
## Models ranked by selectCriteria(x)
##
##
## [[5]]
## [[5]][[1]]
## Formula:          y ~ 1 + Year
## Data: obsdatval
##       AIC       BIC    logLik  df.resid
##  287.8749  335.5911 -131.9375       382
##
## Number of obs: 394
##
## Overdispersion parameter for tweedie family (): 1.41
##
## Tweedie power parameter:  1.5
##
## Fixed Effects:
##
## Conditional model:
## (Intercept)     Year2010     Year2011     Year2012     Year2013     Year2014
##     -4.5296      -0.7751       2.1705       2.0283       2.1592      -0.5905
##    Year2015     Year2016     Year2017     Year2018
##     -0.7361       0.3516      -1.4974       0.2612
##
## [[5]][[2]]
## Global model call: glmmTMB(formula = y ~ season + Year + season:Year, data = obsda
tval,
##     family = TMBfamily, na.action = na.fail, ziformula = ~0,
##     dispformula = ~1)
## ---
## Model selection table
##   cnd((Int)) dsp((Int)) cnd(ssn) cnd(Yer) cnd(ssn:Yer)  AICc   AIC   BIC df
## 1     -4.530          +                +              288.7 287.9 335.6 12
## 2     -4.521          +        +       +              286.3 285.4 337.1 13
## 4     -4.499          +        +       +           +                    22
##     logLik selectCriteria delta
## 1 -131.937          335.6  0.00
## 2 -129.685          337.1  1.47
## 4
## Models ranked by selectCriteria(x)
```

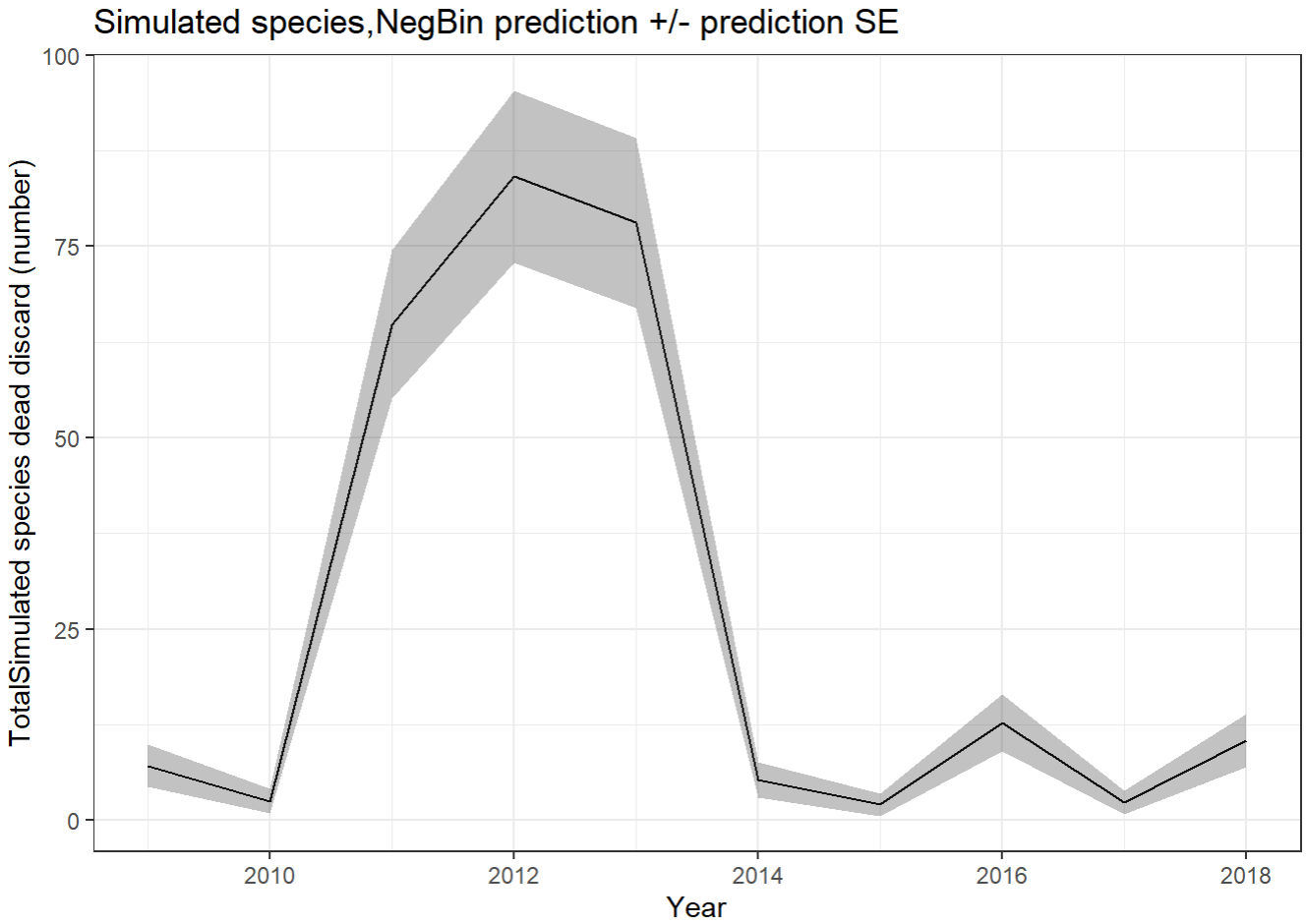
```
 #Make predictions, residuals, etc. for all models
 foreach(mod = 1:length(modelTry)) %do% {
   if(!is.null(modfits[[modelTry[mod]]])) {
     modelSelectTable[[run]]<-modfits[[modelTry[mod]]][[2]]
     modfits[[modelTry[mod]]]<-modfits[[modelTry[mod]]][[1]]
     if(modelTry[mod] %in% c("Lognormal","Gamma")) {
        modfit1<-bin1
        modfit2<-modfits[[modelTry[mod]]]
     } else {
        modfit1<-modfits[[modelTry[mod]]]
        modfit2<-NULL
     }
     if(EstimateBycatch) {
      predvals[[modelTry[mod]]]<-makePredictionsVar(modfit1=modfit1,modfit2=modfit2,m
odType=modelTry[mod],newdat=logdat,printOutput=TRUE)
      plotFits(predvals[[modelTry[mod]]],modelTry[mod],paste0(outVal,"Total",modelTry
[mod],".pdf"))
     }
     if(EstimateIndex) {
      indexvals[[modelTry[mod]]]<-makeIndexVar(modfit1=modfit1,modfit2=modfit2,modTyp
e=modelTry[mod],printOutput=TRUE)
      plotIndex(indexvals[[modelTry[mod]]],modelTry[mod],paste0(outVal,"Index",modelT
ry[mod],".pdf"))
     }
     modelTable[[run]]$formula[mod+1]<-paste(formula(modfits[[modelTry[mod]]]))[[3]]
     temp<-ResidualsFunc(modfits[[modelTry[mod]]],modelTry[mod],paste0(outVal,"Residu
als",modelTry[mod],".pdf"))
     if(!is.null(temp)) {
       residualTab[[run]][,modelTry[mod]]<-temp
       if(residualTab[[run]]["KS.p",modelTry[mod]]<0.01 & ResidualTest) modelFail[ru
n,modelTry[mod]]<-"resid"
     }
     if(is.null(predvals[[modelTry[mod]]])) modelFail[run,modelTry[mod]]<-"cv"
   } else {
     modelFail[run,modelTry[mod]]<-"fit"
   }
 }
```

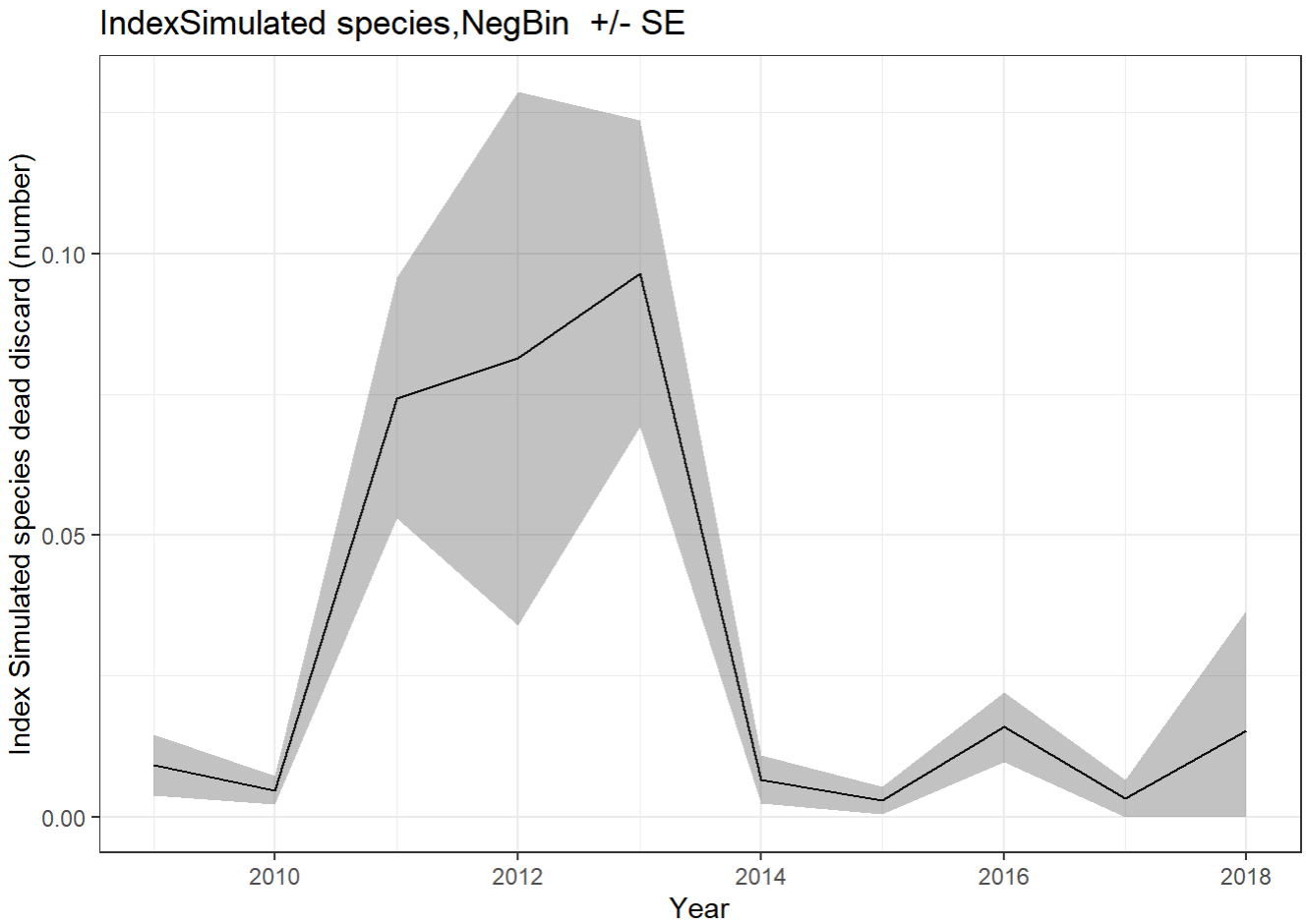

Simulated species,Delta Lognormal prediction +/- prediction SE
TotalSimulated species dead discard (number)
100
75
50
25
0
2010
2012
2014
2016
2018
Year

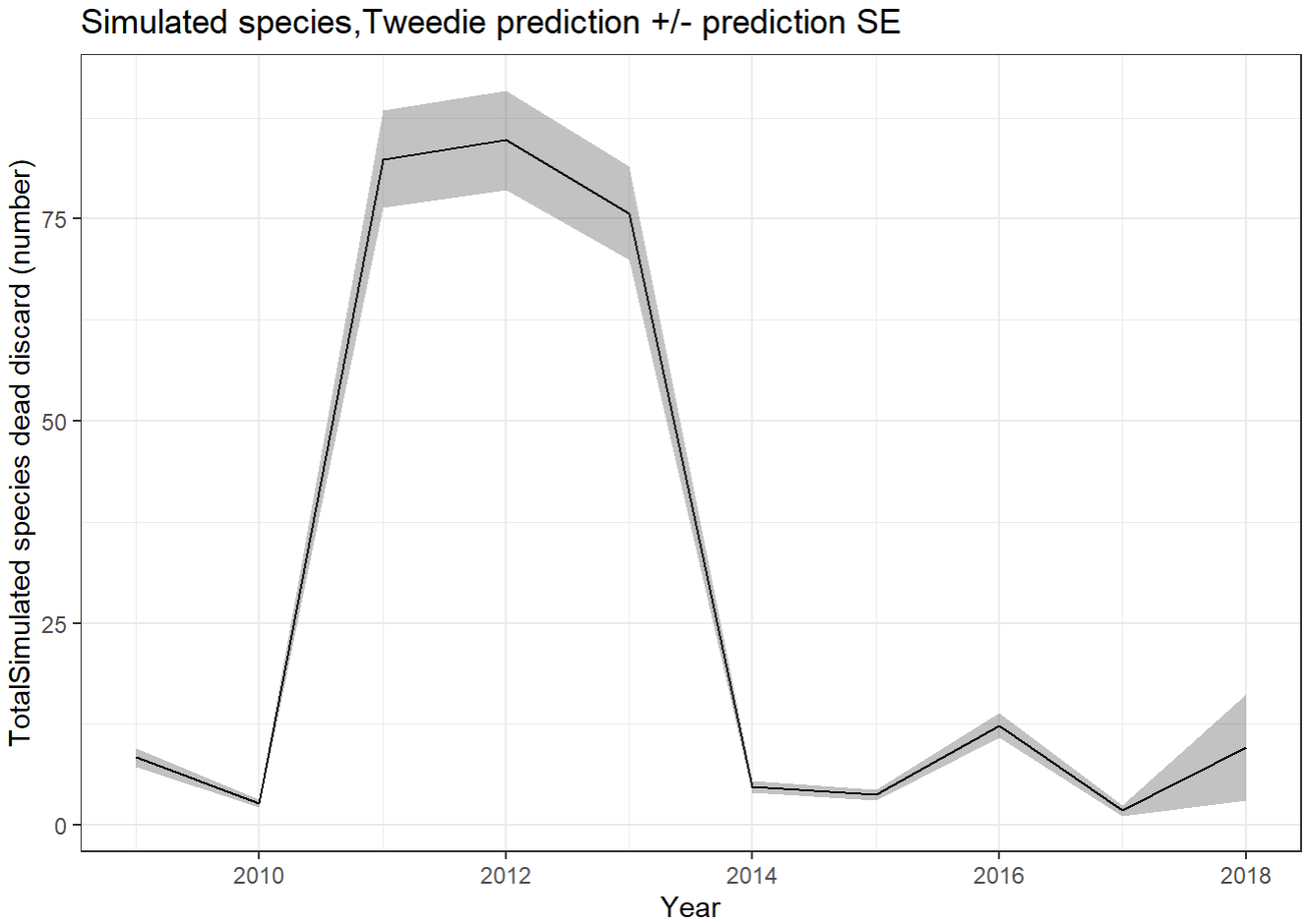

IndexSimulated species,Delta Lognormal +/- SE
Index Simulated species dead discard (number)
0.075
0.050
0.025
0.000
2010
2012
2014
2016
2018
Year

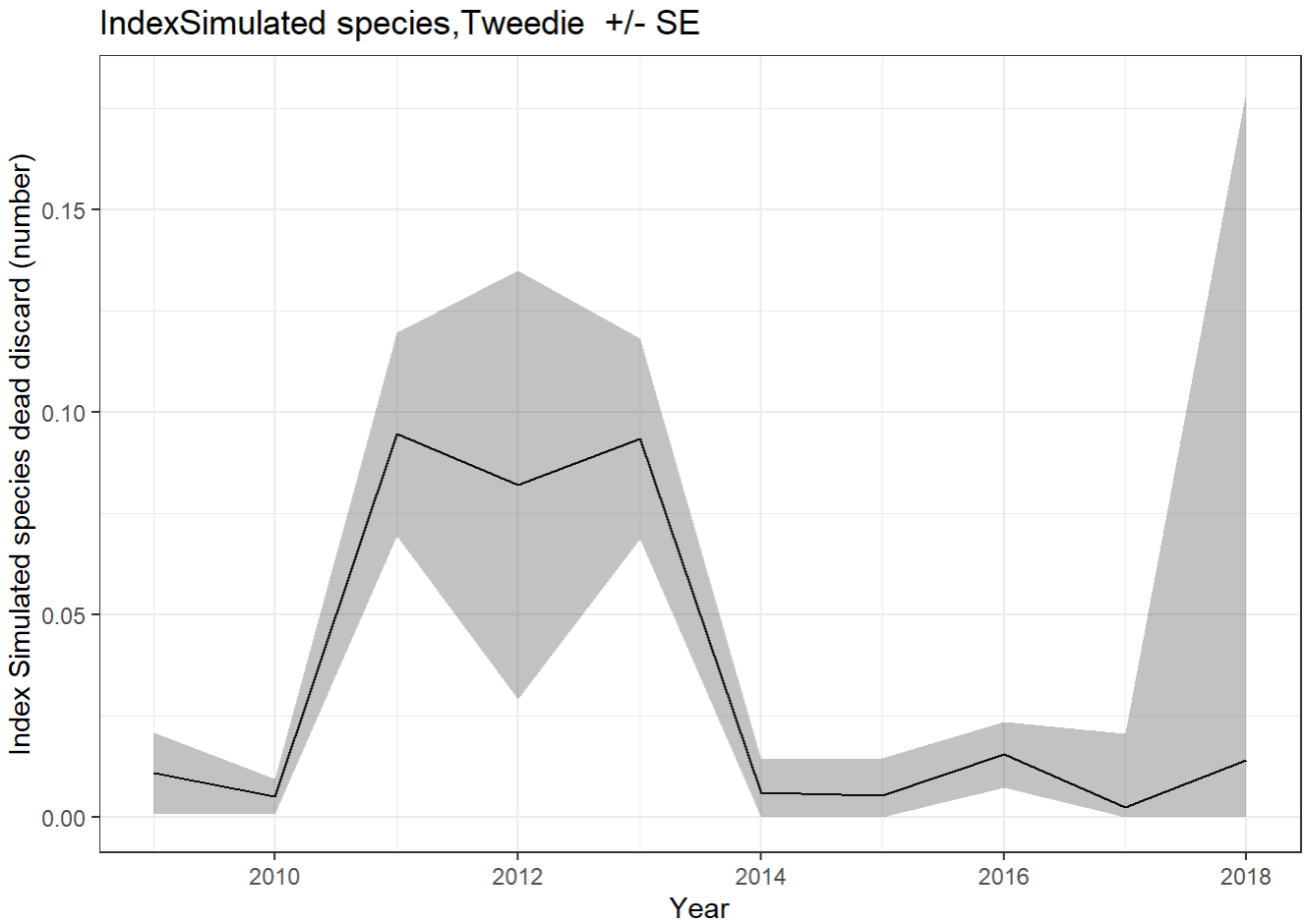

Simulated species,Delta Gamma prediction +/- prediction SE
TotalSimulated species dead discard (number)
100
75
50
25
0
2010
2012
2014
2016
2018
Year

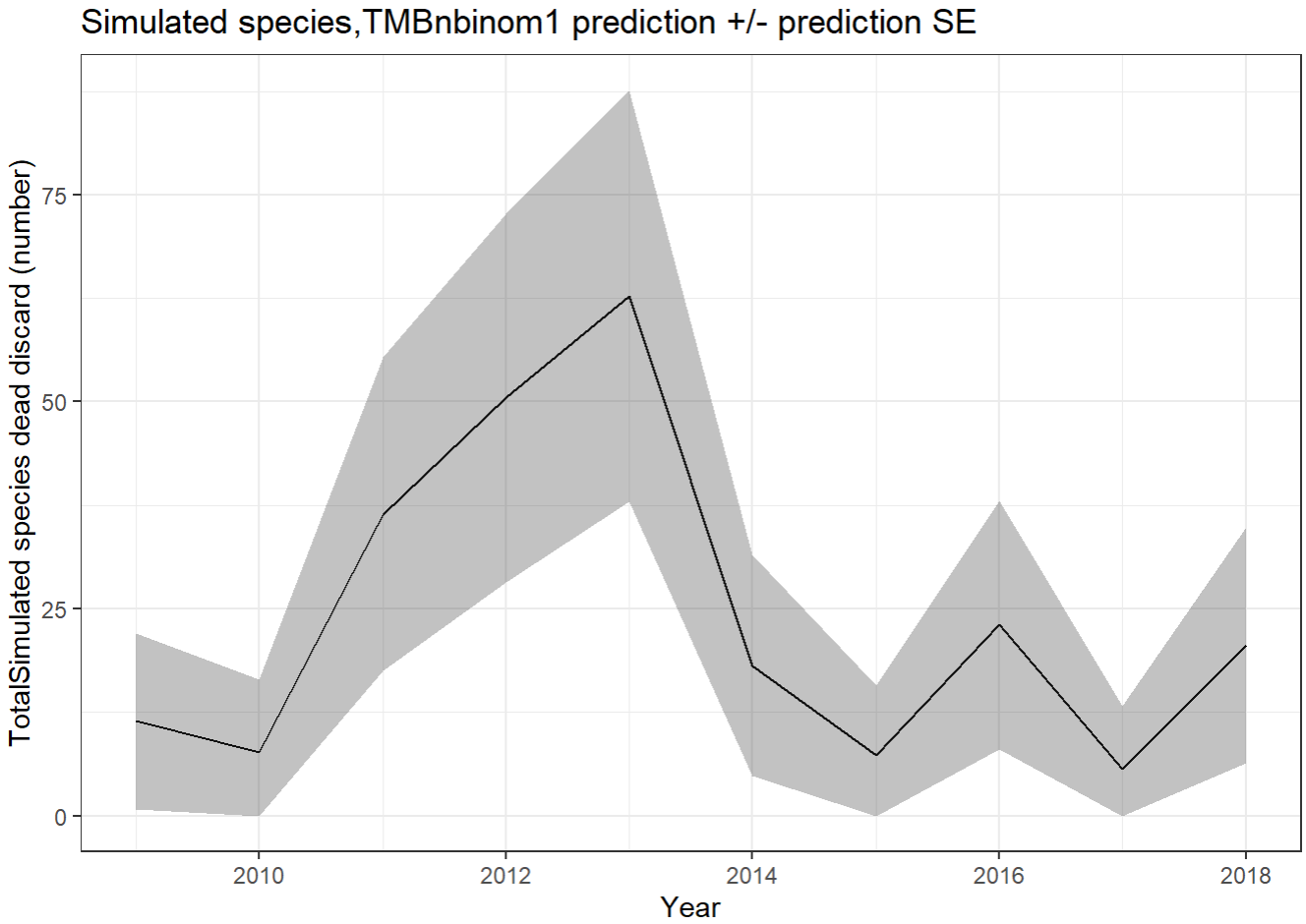

IndexSimulated species,Delta Gamma +/- SE
Index Simulated species dead discard (number)
0.20
0.15
0.10
0.05
0.00
2010
2012
2014
2016
2018
Year

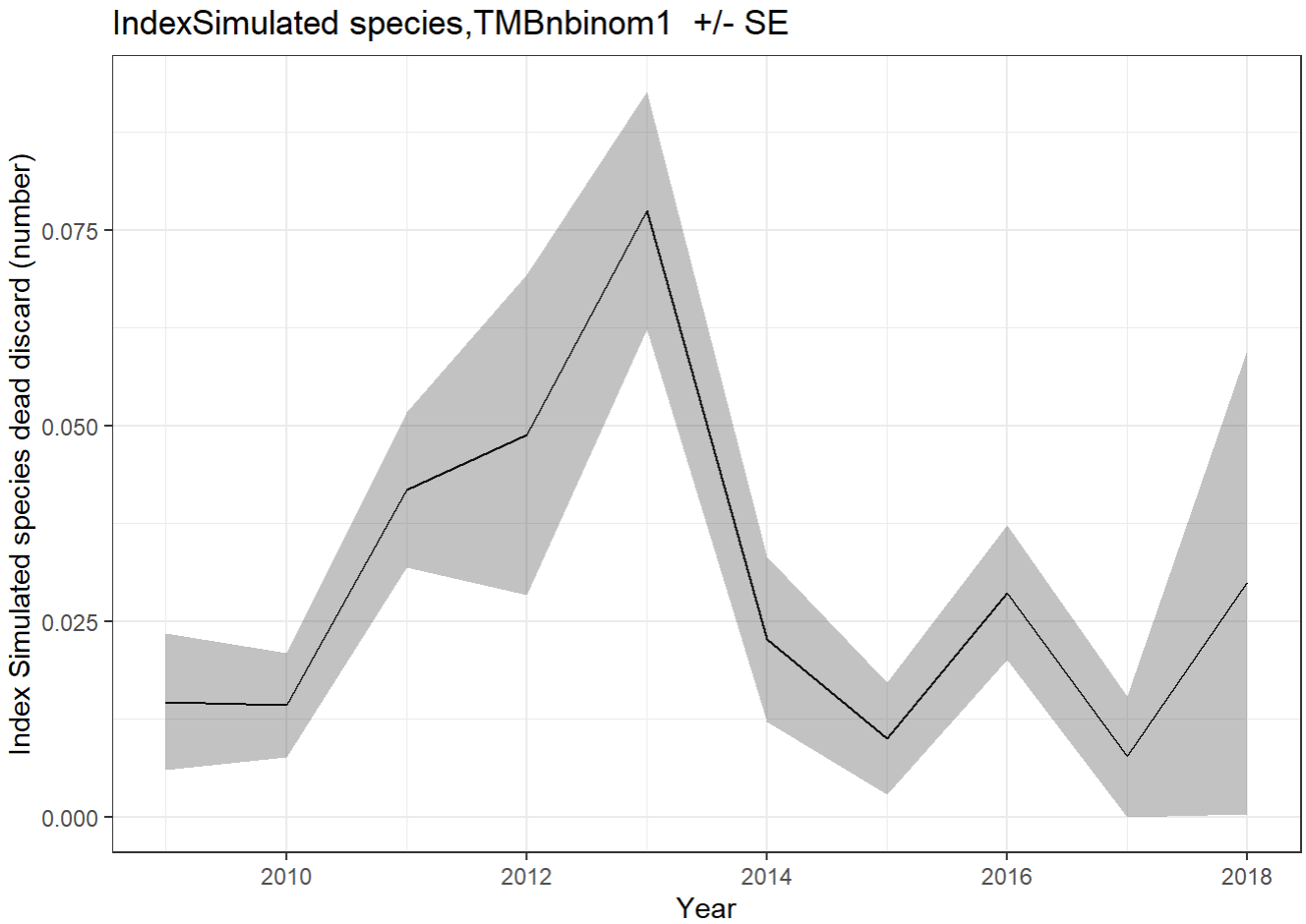

Simulated species,NegBin prediction +/- prediction SE
TotalSimulated species dead discard (number)
100
75
50
25
0
2010
2012
2014
2016
2018
Year

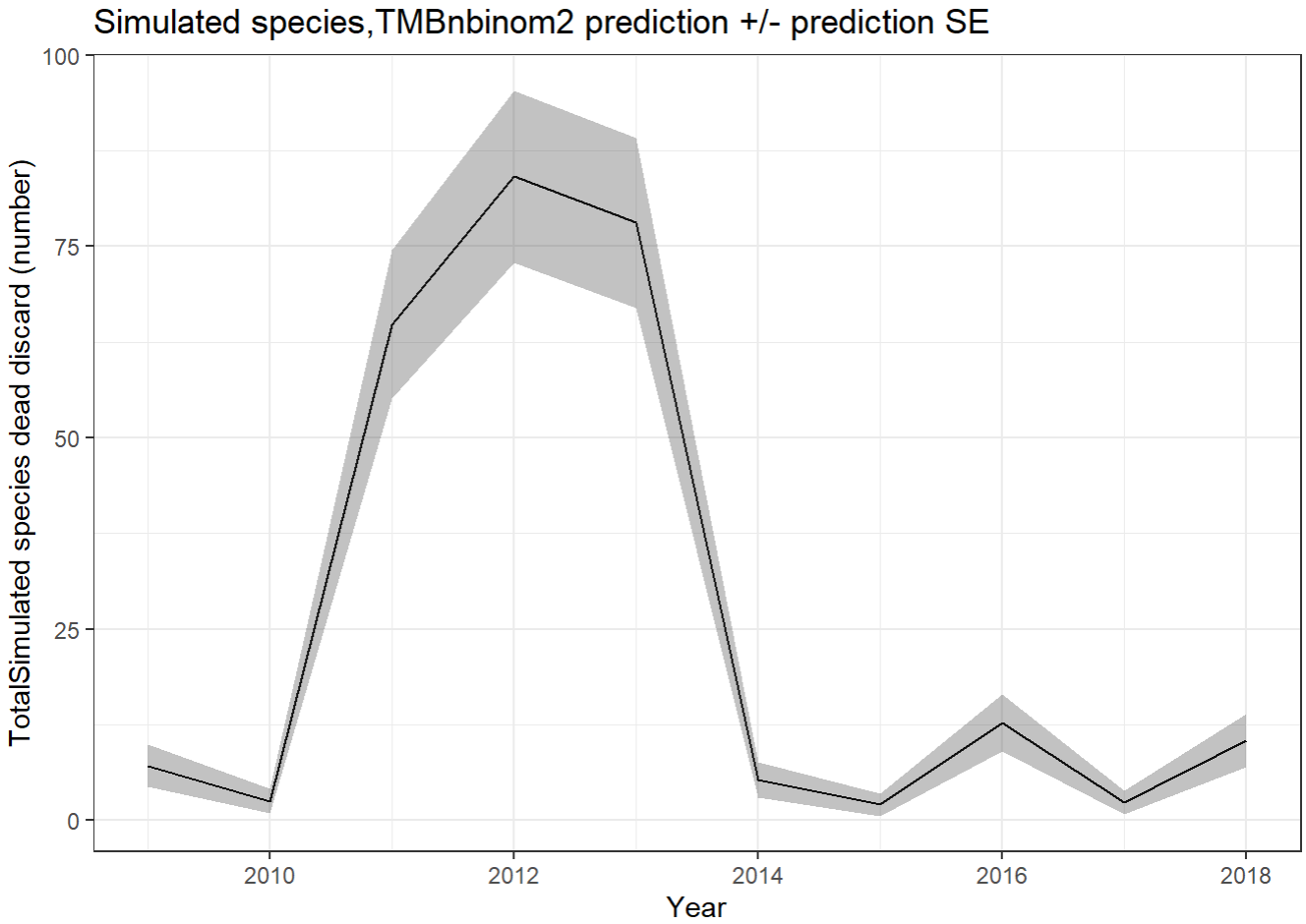

IndexSimulated species,NegBin +/- SE
Index Simulated species dead discard (number)
0.10
0.05
0.00
2010
2012
2014
2016
2018
Year

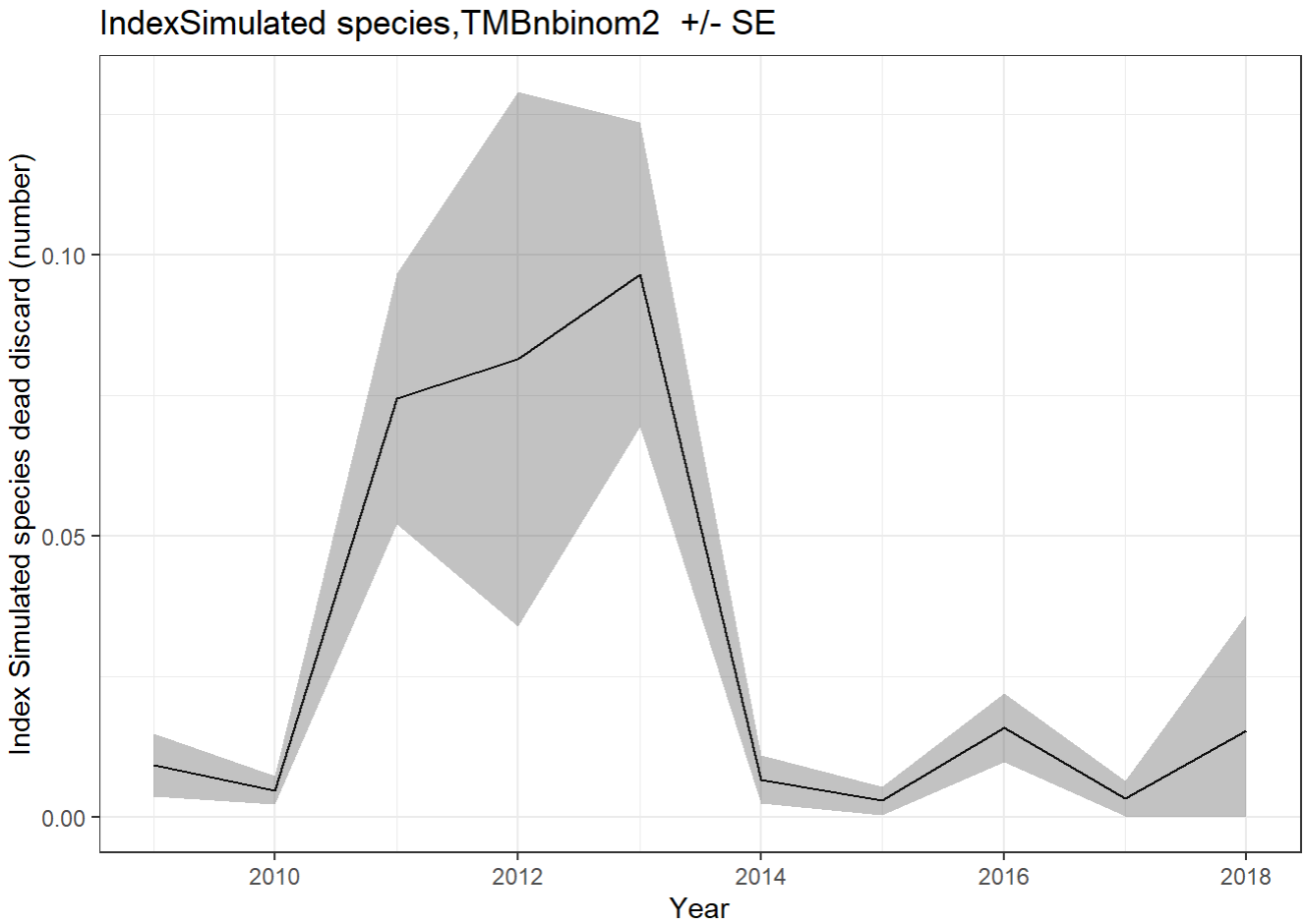

Simulated species,Tweedie prediction +/- prediction SE
TotalSimulated species dead discard (number)
0
25
50
75
2010
2012
2014
2016
2018
Year

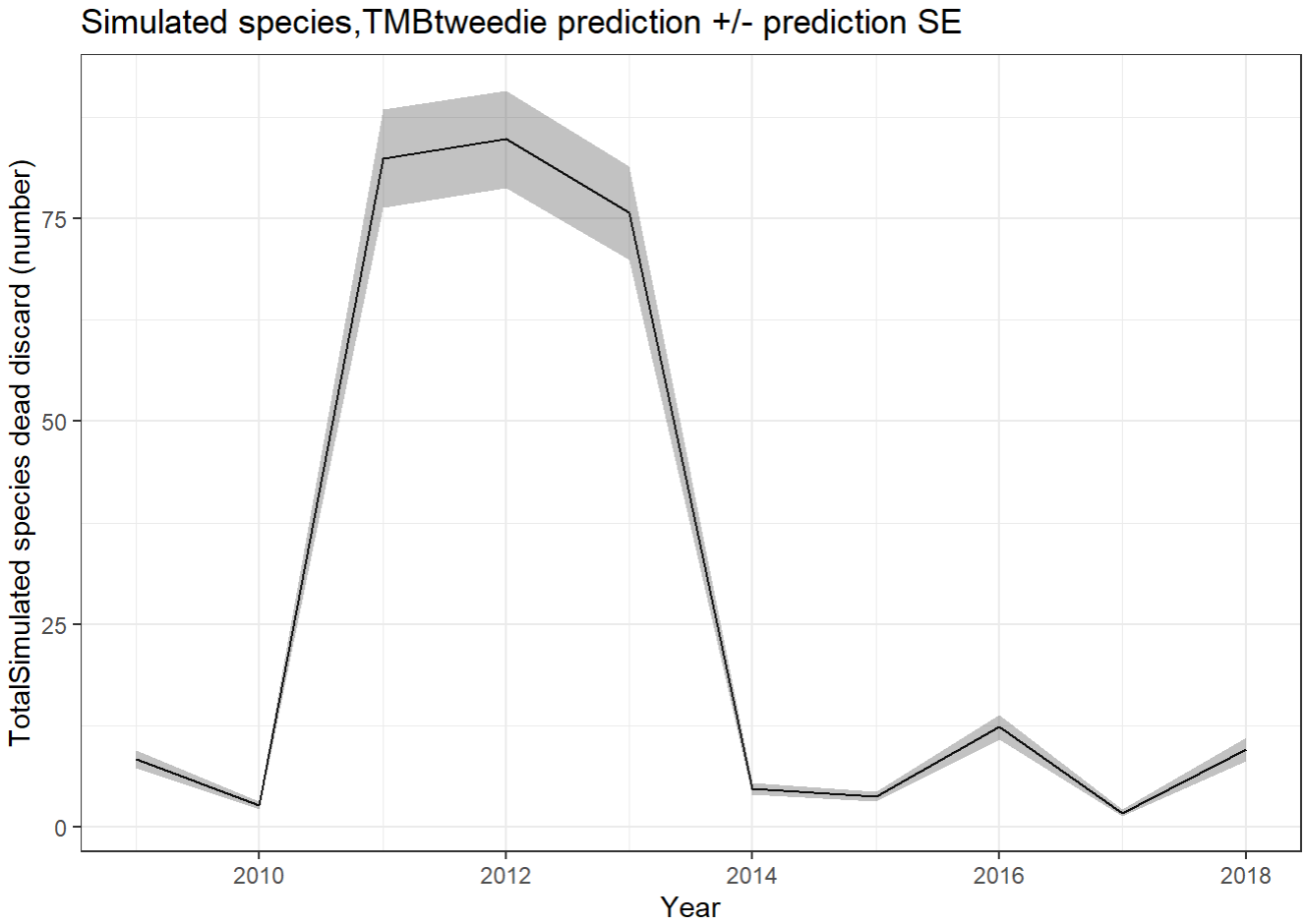

IndexSimulated species,Tweedie +/- SE
Index Simulated species dead discard (number)
0.00
0.05
0.10
0.15
2010
2012
2014
2016
2018
Year

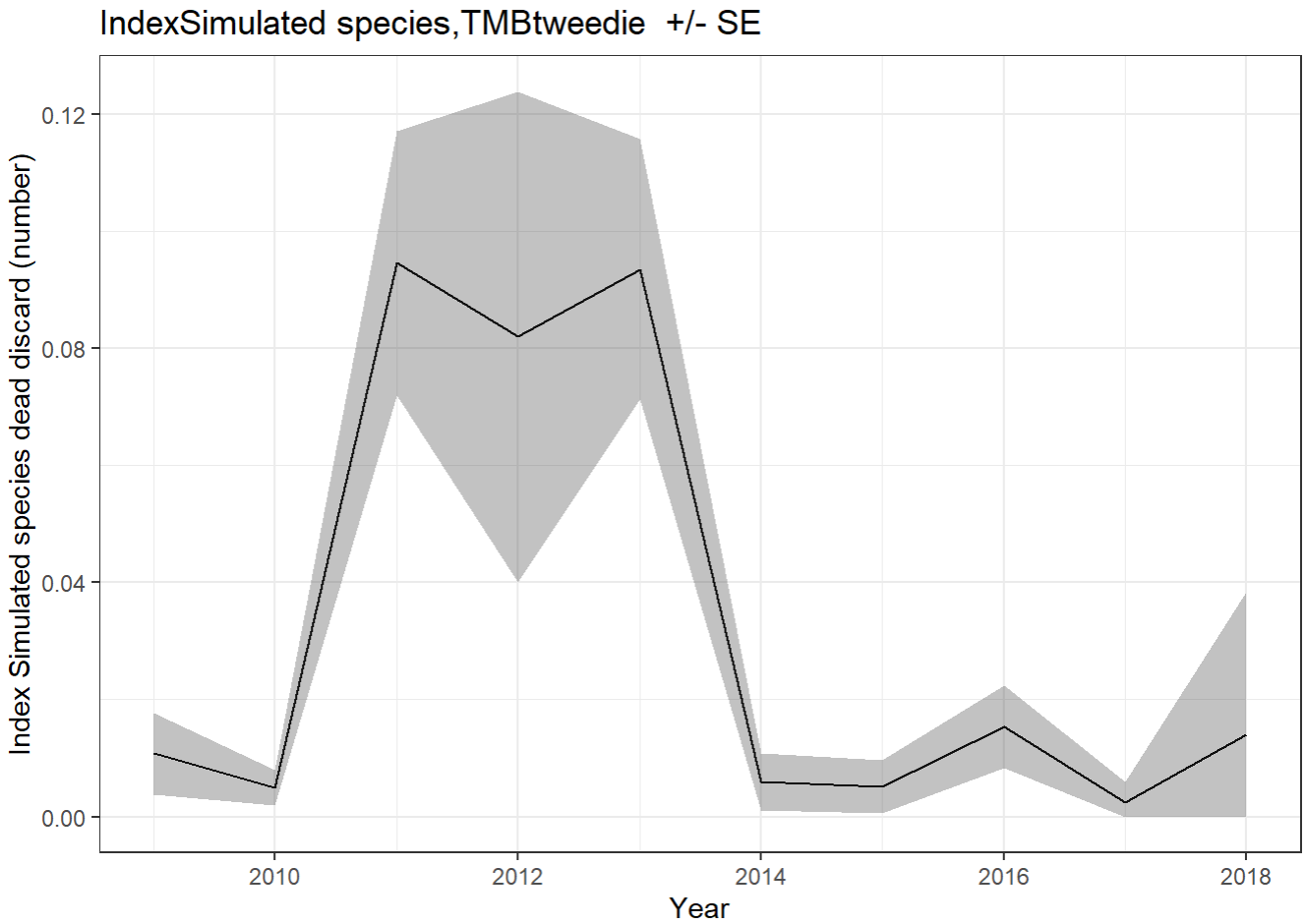

Simulated species,TMBnbinom1 prediction +/- prediction SE
TotalSimulated species dead discard (number)
75
50
25
0
2010
2012
2014
2016
2018
Year


IndexSimulated species,TMBnbinom1 +/- SE
Index Simulated species dead discard (number)
0.075
0.050
0.025
0.000
2010
2012
2014
2016
2018
Year


Simulated species,TMBnbinom2 prediction +/- prediction SE
TotalSimulated species dead discard (number)
100
75
50
25
0
2010
2012
2014
2016
2018
Year


IndexSimulated species,TMBnbinom2 +/- SE
Index Simulated species dead discard (number)
0.10
0.05
0.00
2010
2012
2014
2016
2018
Year

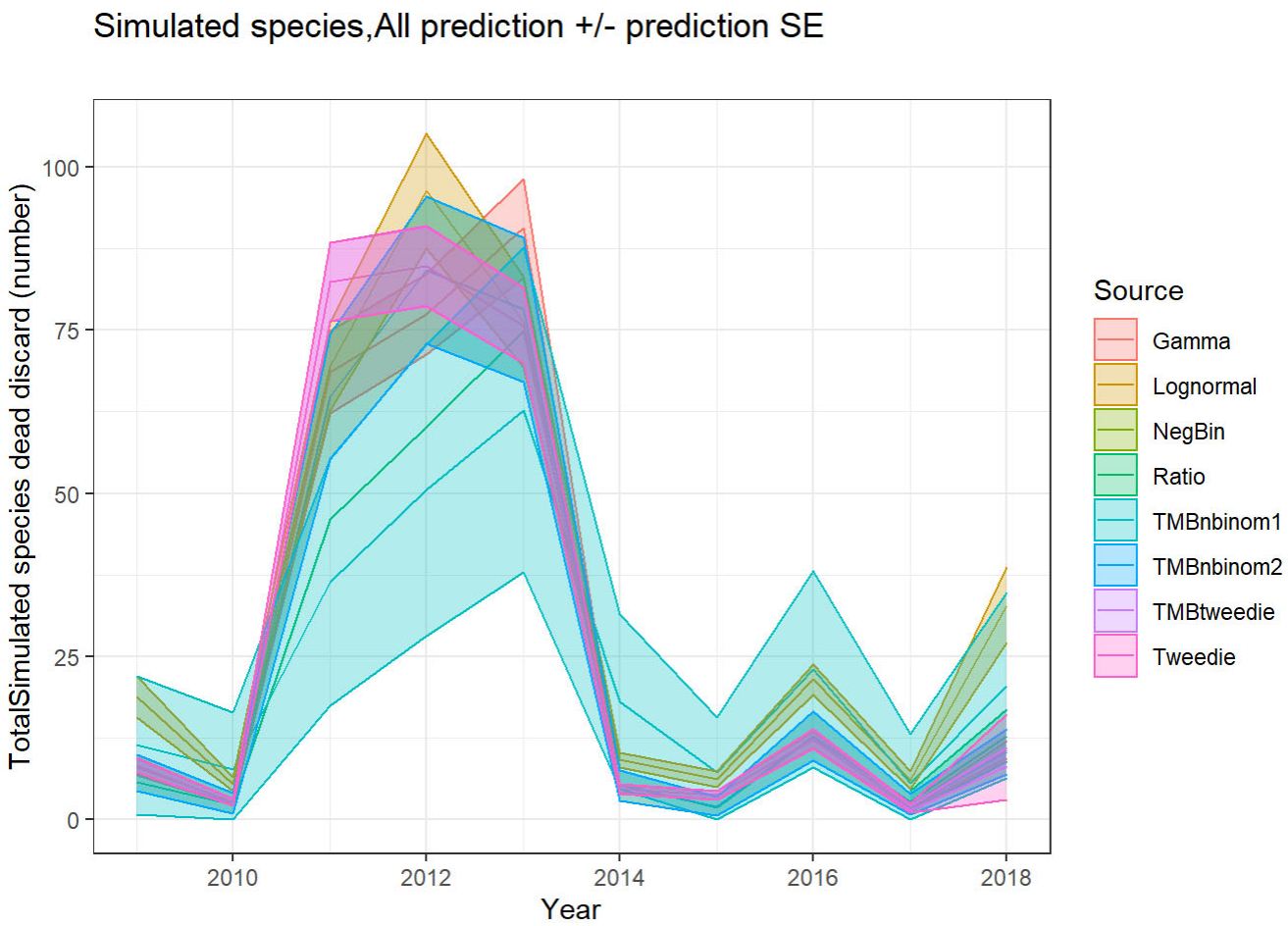

Simulated species,TMBtweedie prediction +/- prediction SE
TotalSimulated species dead discard (number)
0
25
50
75
2010
2012
2014
2016
2018
Year


IndexSimulated species,TMBtweedie +/- SE
Index Simulated species dead discard (number)
0.00
0.04
0.08
0.12
2010
2012
2014
2016
2018
Year

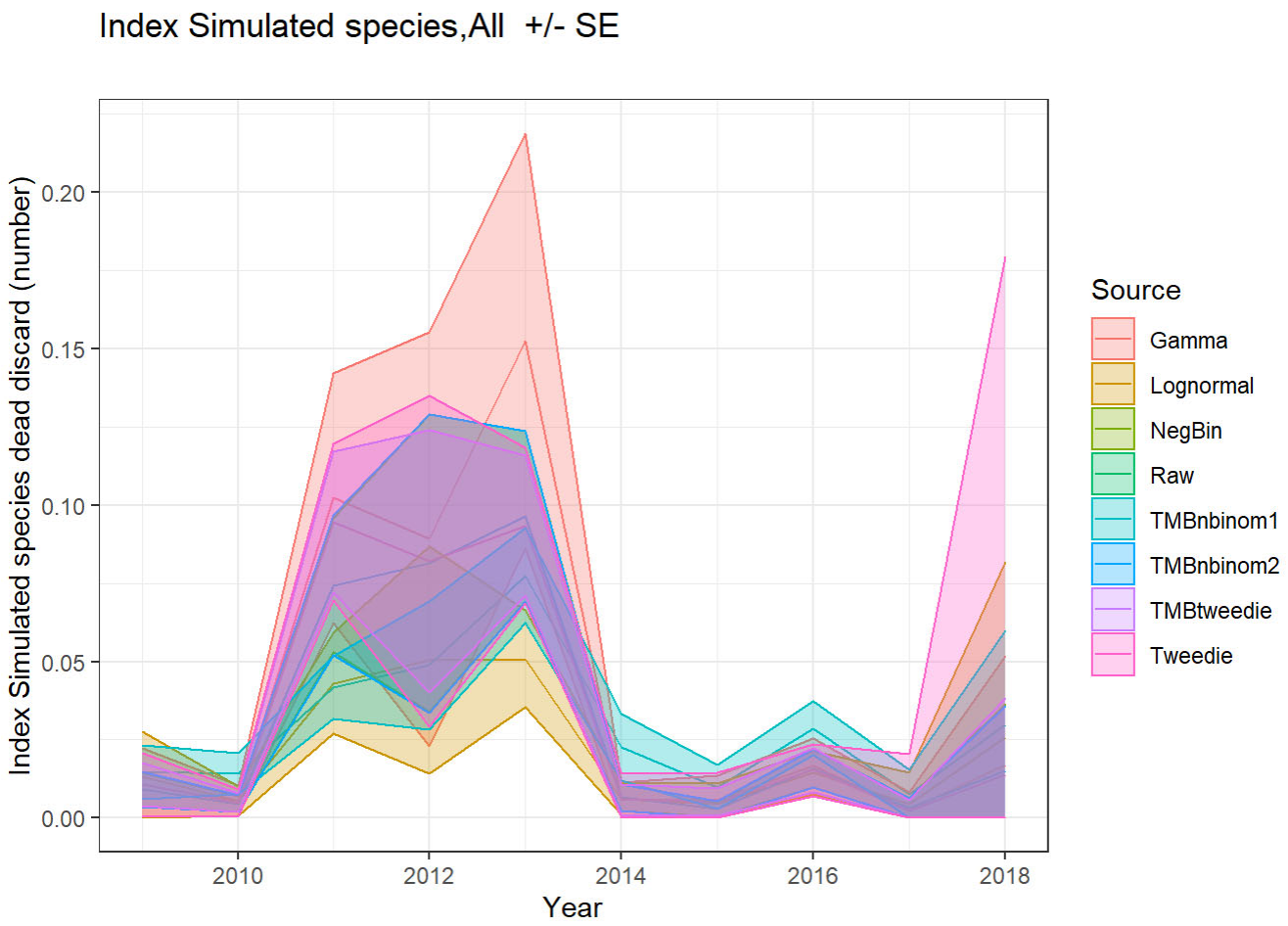
```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
```

The next part compares the predictions of total catch for all models that converged adequately and had CVs in predicted catch less than 10. The predictions from an unstratified ratio estimator are shown for comparison. However, this is not expected to be the same as the model based estimates if any of the predictor variables (other than year) are important. To use a ratio estimator to make a valid design-based estimate of total bycatch would require a sufficient sample size in each combination of the predictor variables to estimate bycatch in each stratum with the ratio estimator. A table summarizing the DHARMa diagnostics is also printed (Hartig (2020)). 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.

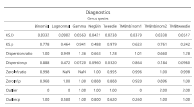
```
  #Compare all predictions
 if(EstimateBycatch) {
  yearsumgraph<-yearSum[[run]] %>% dplyr::select(Year=Year,Total=CatEst,Total.se=Cats
e) %>%
    mutate(TotalVar=Total.se^2,Total.cv=Total.se/Total,TotalFixed=Total)
  allmods[[run]]<-bind_rows(c(predvals,list(Ratio=yearsumgraph)),.id="Source")
  allmods[[run]]$Valid<-ifelse(modelFail[run,match(allmods[[run]]$Source,dimnames(mod
elFail)[[2]])]=="-" | allmods[[run]]$Source=="Ratio",1,0)
  plotFits(allmods[[run]],modType="All",paste0(outVal,"AllTotal.pdf"))
 }
```




```
 if(EstimateIndex) {
  x<-indexvals
  for(i in 1:length(x)) x[[i]]=dplyr::select(x[[i]],Year,Index, SE)
  x<-c(x,list(Raw=yearSum[[run]] %>% mutate(Index=CPUE, SE=NA,Year=factor(as.characte
r(Year))) %>% dplyr::select(Year,Index,SE)))
  allindex[[run]]<-bind_rows(x,.id="Source")
  allindex[[run]]$Valid<-ifelse(modelFail[run,match(allindex[[run]]$Source,dimnames(m
odelFail)[[2]])]=="-" | allindex[[run]]$Source=="Raw",1,0)
  plotIndex(allindex[[run]],modType="All",paste0(outVal,"AllIndex.pdf"))
 }
```




```
  #Show the diagnostic table
  printTableFunc("Diagnostics",sp[run],residualTab[[run]],paste0(outVal,"residualDiag
nostics.pdf"),useRowNames = TRUE)
```



```
  write.csv(residualTab[[run]],paste0(outVal,"residualDiagnostics.csv"))
```

The next part runs the cross validation if requested. Only observation error models that converged and produced reasonable results with the complete dataset 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. 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.

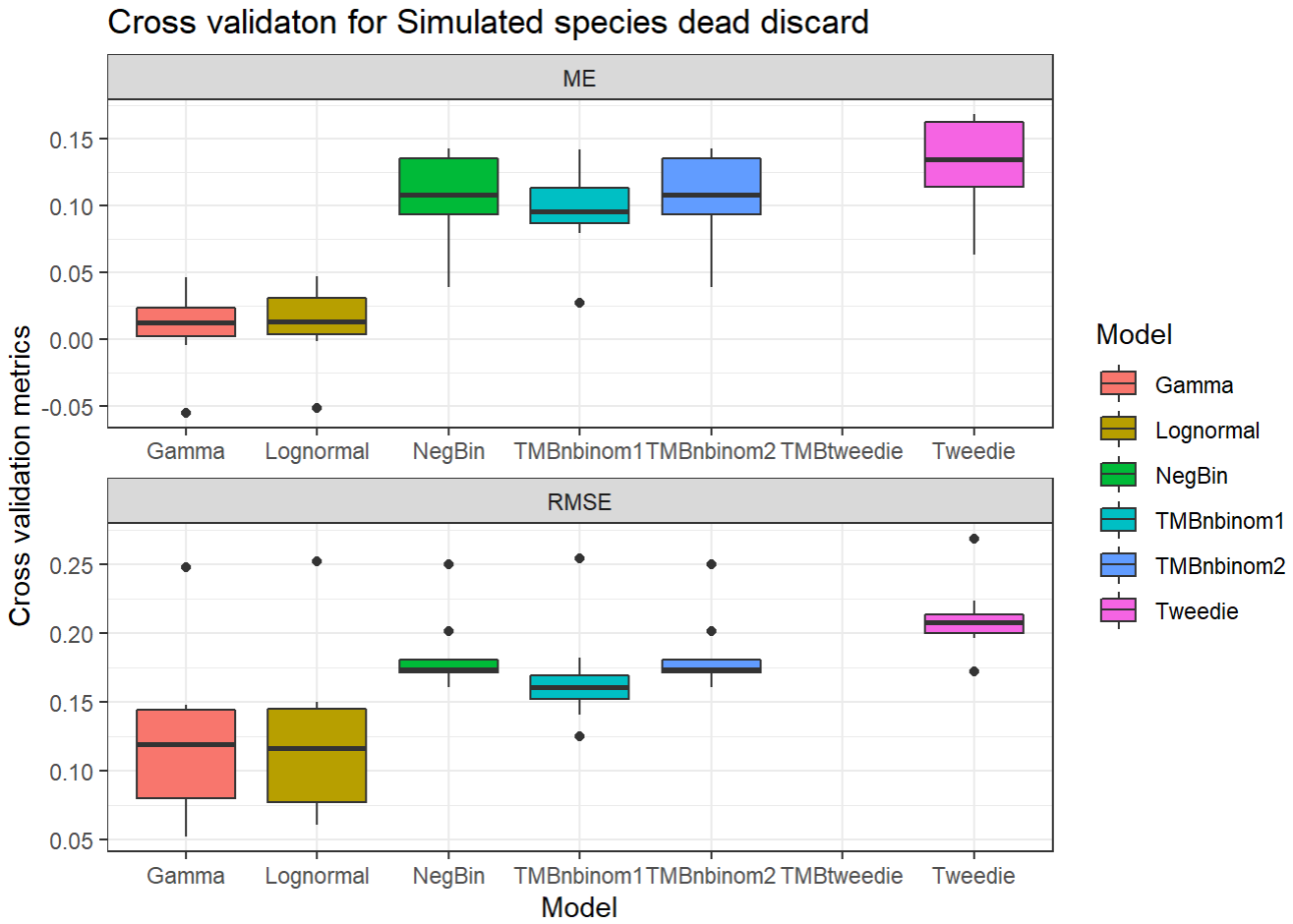
$$ME = \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i - y_i$$

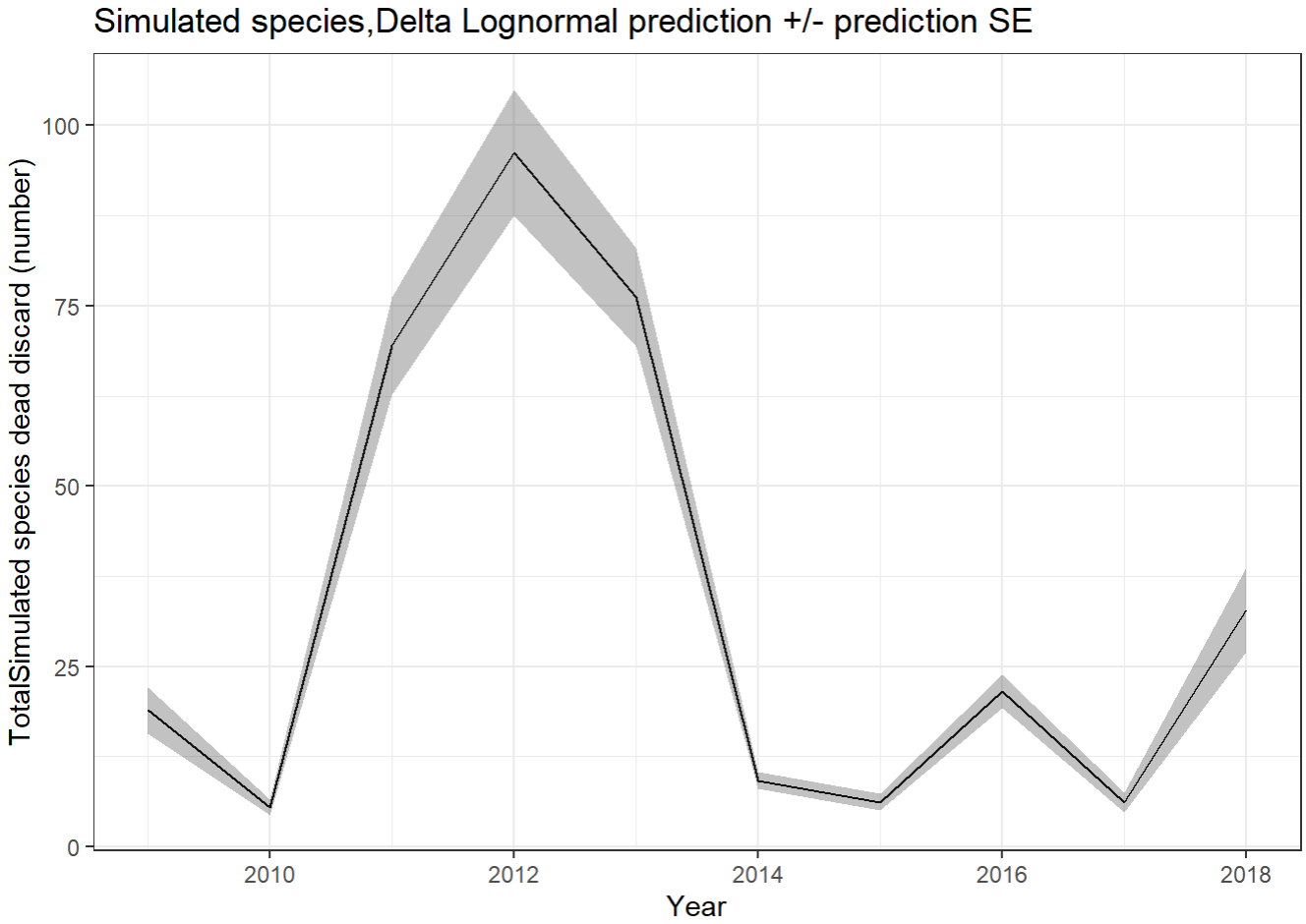
The output from this section is tables showing the RMSE and MAE values for each fold for each type of model, and boxplots of the RMSE and MAE values across the folds for each method.

```
 ######## Cross validation 10 fold  ###################################
  if(DoCrossValidation & length(which(modelFail[run,-1]=="-"))>0) {  #Don't do unless
at least one model worked
  if(NumCores>3)  {
    cl<-makeCluster(NumCores-2)
    registerDoParallel(cl)
  }
  datval$cvsample<-sample(rep(1:10,length=dim(datval)[1]),replace=FALSE)
  table(datval$cvsample,datval$Year)
  rmsetab[[run]]<-data.frame(matrix(NA,10,length(modelTry),dimnames=list(1:10,modelTr
y)))
  metab[[run]]<-rmsetab[[run]]
  foreach(i=1:10 ) %do% {
   datin<-datval[datval$cvsample!=i,]
   datout<-datval[datval$cvsample==i,]
   datout$SampleUnits<-rep(1,dim(datout)[1])
   if(DredgeCrossValidation) bin1<-findBestModelFunc(datin,"Binomial")[[1]] else
    bin1<-FitModelFuncCV(formula(paste0("y~",modelTable[[run]]$formula[1])),modType="
Binomial",obsdatval=datin)
   if("Lognormal" %in% modelTry | "Gamma" %in% modelTry) {
     posdat<-filter(datin,pres==1)
     for(mod in which(modelTry %in% c("Lognormal","Gamma"))) {
       if(modelFail[run,modelTry[mod]]=="-" & min(summary(posdat$Year))>0) {
         if(DredgeCrossValidation) modfit1<-findBestModelFunc(posdat,modelTry[mo
d])[[1]] else
          modfit1<-FitModelFuncCV(formula(paste0("y~",modelTable[[run]]$formula[mod+
1])),modType=modelTry[mod],obsdatval=posdat)
         predcpue<-makePredictions(bin1,modfit1,modelTry[mod],datout)
         rmsetab[[run]][i,modelTry[mod]]<-getRMSE(predcpue$est.cpue,datout$cpue)
         metab[[run]][i,modelTry[mod]]<-getME(predcpue$est.cpue,datout$cpue)
       }
     }
   }
   for(mod in which(!modelTry %in% c("Lognormal","Gamma"))) {
     if(modelFail[run,modelTry[mod]]=="-") {
       if(DredgeCrossValidation) modfit1<-findBestModelFunc(datin,modelTry[mod])[[1]]
else
        modfit1<-FitModelFuncCV(formula(paste0("y~",modelTable[[run]]$formula[mod+
1])),modType=modelTry[mod],obsdatval=posdat)
       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)
     }
   }
  }
 if(NumCores>3) stopCluster(cl)
# Calculate RMSE and ME
 modelTable[[run]]$RMSE[-1]<-apply(rmsetab[[run]],2,mean,na.rm=TRUE)
 modelTable[[run]]$ME[-1]<-apply(metab[[run]],2,mean,na.rm=TRUE)
 printTableFunc("Run summary",sp[run],modelTable[[run]],paste0(outVal,"modelSummary.p
df"),useRowNames = FALSE)
```

```
 write.csv(residualTab[[run]],paste0(outVal,"modelSummary.csv"))
 write.csv(rmsetab[[run]],paste0(outVal,"rmse.csv"))
 write.csv(metab[[run]],paste0(outVal,"me.csv"))
 plotCrossVal(rmsetab[[run]],metab[[run]],paste0(outVal,"CrossValidation.pdf"))
 #Select best model based on cross validation
 best<-which(!is.na( modelTable[[run]]$RMSE) &
     modelTable[[run]]$RMSE==min(modelTable[[run]]$RMSE,na.rm=TRUE))-1 #Less 1 to exc
lude binomial
 bestmod[run]<-modelTry[best]
 predbestmod[[run]]<-predvals[[modelTry[best]]]
 plotFits(predbestmod[[run]],bestmod[run],paste0(outVal,"BestTotal.pdf"))
 modelTable[[run]]
 }
```

```
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = objec
t$xlevels) :
##   factor Year has new levels 2018
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = xlevel
s) :
##   factor Year has new levels 2018
## Error in c1[rownames(c2), colnames(c2), drop = FALSE] :
##   subscript out of bounds
## Error in c1[rownames(c2), colnames(c2), drop = FALSE] :
##   subscript out of bounds
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = objec
t$xlevels) :
##   factor Year has new levels 2017
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = xlevel
s) :
##   factor Year has new levels 2017
## Error in c1[rownames(c2), colnames(c2), drop = FALSE] :
##   subscript out of bounds
## Error in c1[rownames(c2), colnames(c2), drop = FALSE] :
##   subscript out of bounds
## Error in model.frame.default(data = obsdatval, drop.unused.levels = TRUE,  :
##   variable lengths differ (found for 'Year')
```


Cross validaton for Simulated species dead discard
ME
RMSE
Cross validation metrics
0.15
0.10
0.05
0.00
-0.05
0.25
0.20
0.15
0.10
0.05
Gamma
Lognormal
NegBin
TMBnbinom1
TMBnbinom2
TMBtweedie
Tweedie
Model
Model
Gamma
Lognormal
NegBin
TMBnbinom1
TMBnbinom2
Tweedie


Simulated species,Delta Lognormal prediction +/- prediction SE
TotalSimulated species dead discard (number)
100
75
50
25
0
2010
2012
2014
2016
2018
Year

```
##        model                          formula      RMSE          ME
## 1   Binomial                        1 + Year        NA          NA
## 2  Lognormal                        1 + Year 0.1229146 0.011858576
## 3      Gamma              season + 1 + Year 0.1234997 0.008492528
## 4     NegBin 1 + Year + offset(log(Effort)) 0.1843784 0.106633503
## 5    Tweedie                        1 + Year 0.2109734 0.130642134
## 6 TMBnbinom1 1 + Year + offset(log(Effort)) 0.1681755 0.096043984
## 7 TMBnbinom2 1 + Year + offset(log(Effort)) 0.1843784 0.106633516
## 8 TMBtweedie                        1 + Year       NaN         NaN
```

```
rm(list=c("bin1","predvals","modfits"))
print(paste(run, common[run],"complete"))
```

```
## [1] "1 Simulated species complete"
```

```
#}
source(paste0(baseDir,"/5.FinalModelPrinting.r"))
###################### End of analysis loop ######################

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

At the end of the analysis loop, summaries are printed to the main output directory. They are: (1) bestTotal.pdf, which shows the annual bycatch estimates for the “best” model for all the bycatch species or disposition groups that were run in the loop, (2) crossval.pdf which shows the crossvalidation figure, (3) Datasummary.pdf which shows the data summaries, (4) modelfail.pdf and modelfail.csv which show the performance criteria for each model for each species, (5) bestIndex.pdf showing the annual abundance index if requested (6) files labeled with the name of each species or disposition category in the loop showing the complete outputs for that group, and (7) .rData file containing the complete analysis. More .pdf and .csv files are available in separate folders for each group.

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, you can do that in the file called 6.ExamineModelResults.

Looking at a particular model run

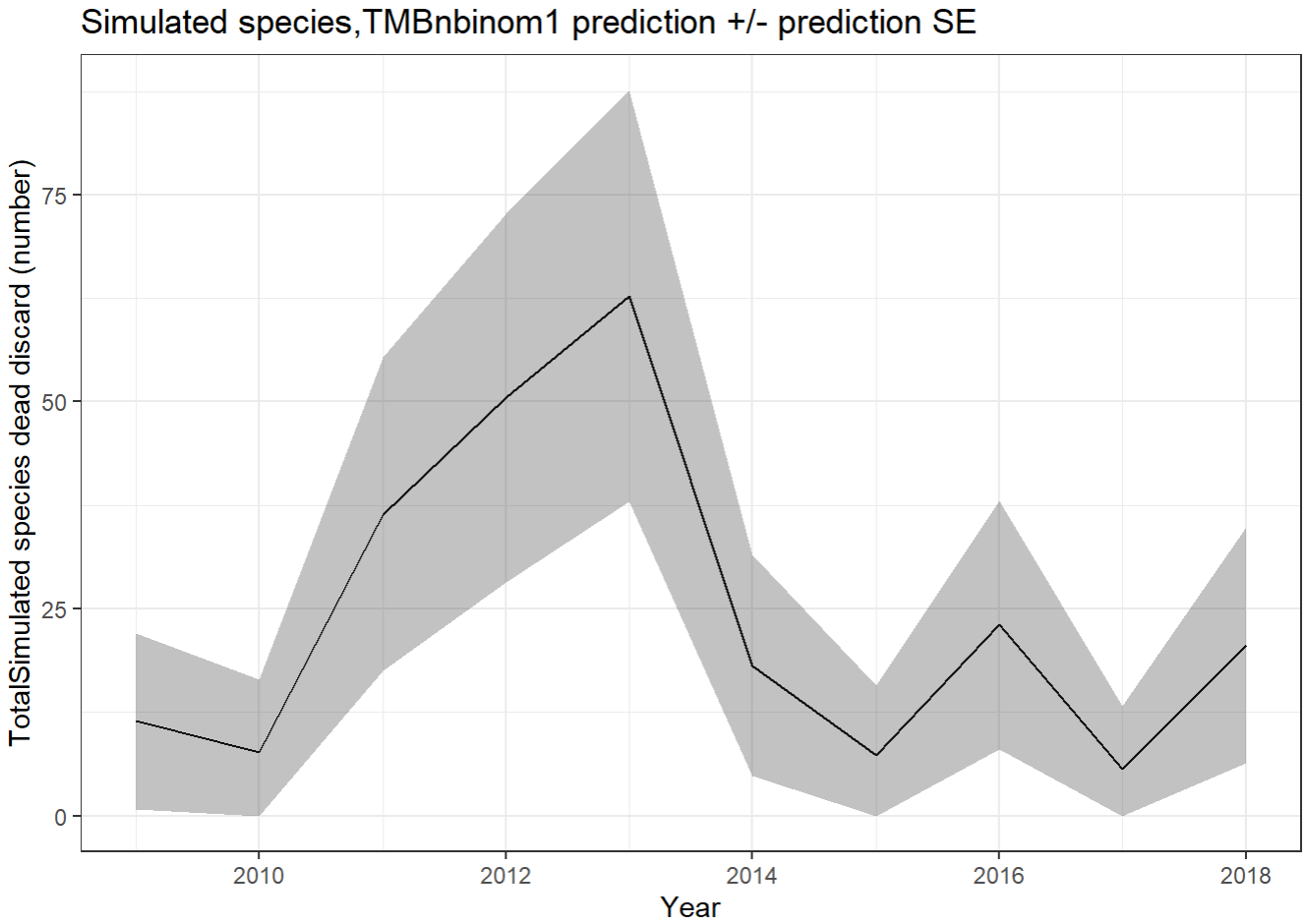
The file called 6.ExamineModelResults allows the user to specify a particular model, and print out all its summary statistics and diagnostics. This is useful if you want to choose a model other than the RMSE best model and look at it more closely. The output will be in a separate directory including the word “selected”

```
#This code allows the user to select a particular model and look at its outputs

# If the model results are not currently loaded, you may load them by
# uncommenting the following lines.
#specFile<-"C:/Users/ebabcock/Dropbox/bycatch project/Current R code/1.BycatchModelSpecification.r"
#source(specFile)
# load("C:/Users/ebabcock/Dropbox/bycatch project/Current R code/OutputSimulated data example/R.workspace.rData")

#Select the model you want to run
#Options for model types are: c("Lognormal","Gamma","NegBin","Tweedie","TMBnbinom1","TMBnbinom2", "TMBtweedie")
ModelSelect<-"TMBnbinom1"
#Specify a model formula for the model, or for the positive catch portion for delta models
FormulaSelect<- as.formula(y~Year)
#Specify a model for the binomial portion of delta models (Not used for others)
FormulaBin<- as.formula(y~Year)
#Give the number of the species to pull out the correct data
SpeciesSelect<-1

# No changes from here
datval<-dat[[SpeciesSelect]]
selectOutDir<-paste0(outDir,"/",common[SpeciesSelect],catchType[SpeciesSelect],"Selected")
if(!dir.exists(selectOutDir)) dir.create(selectOutDir)
modlist<-FitModelFunc(FormulaBin,FormulaSelect,ModelSelect,datval,selectOutDir)
predvals<-makePredictionsVar(modlist[[1]],modlist[[2]],modType=ModelSelect,newdat=logdat)
write.csv(predvals,paste0(selectOutDir,"/",ModelSelect,"AnnualSummary.csv"))
plotFits(predvals,ModelSelect,paste0(selectOutDir,"/",ModelSelect,"SelectedModel.pdf"))
```




```
if(ModelSelect %in% c("Lognormal","Gamma")) ResidualsFunc(modlist[[1]],"Binomial",pas
te0(selectOutDir,"/",ModelSelect,"ResidualsBin.pdf")) else
 ResidualsFunc(modlist[[1]],ModelSelect,paste0(selectOutDir,"/",ModelSelect,"Residual
s.pdf"))
```

```
##             KS.D             KS.p Dispersion.ratio     Dispersion.p
##       0.03791006       0.62297124       1.00516466       0.86400000
##    ZeroInf.ratio        ZeroInf.p          Outlier        Outlier.p
##       0.99553478       0.92000000       0.00000000       0.16000000
```

```
if(!is.null(modlist[[2]])) ResidualsFunc(modlist[[2]],ModelSelect,paste0(selectOutDi
r,"/",ModelSelect,"Residuals.pdf"))
```

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 (https://CRAN.R-project.org/package=gridExtra).

Barton, Kamil. 2020. *MuMIn: Multi-Model Inference*. https://CRAN.R-project.org/package=MuMIn (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 (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 (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 (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 (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 (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 (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 10.1139/F92-278 (https://doi.org/Doi 10.1139/F92-278).

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

RStudio Team. 2020. *RStudio: Integrated Development Environment for r*. Boston, MA: RStudio, PBC. http://www.rstudio.com/ (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 (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/ (http://www.jstatsoft.org/v21/i12/).

———. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org (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 (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 (https://CRAN.R-project.org/package=gtable).

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

Models.”