Bycatch Estimator User Guide

Elizabeth A. Babcock

2021-02-08

ebabcock@rsmas.miami.edu

Introduction

The R code called BycatchFunctions.r runs a generic model-based bycatch estimation procedure. 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, 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); 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. The best model according to cross validation is the one with the lowest root mean square error (RMSE) in the predicted CPUE. 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 direcly 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 postive 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 to get the total catch in each year. Because catch is being predicted in each trip, the variance of the prediciton 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 logbok data is aggregated across multiple trips the effort is allocated equally to all the trips in a row of the database for the purpose of simulating catches. This allocation procedure is not needed to estimate the mean catches, but it is necesary to estimate the variances correctly.

The model can estimate bycatch 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.

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 code at the top of the .r file. 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 trips, 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 observer data, which is equal to 1 if the data are not aggregated, or the number of trips represented by each row if the data are aggregated. This is needed to predict catches by trip for the variance calculations. Finally, include a column in the observer data for the amount of unsampled effort in each trip. This can be zero if all trips are 100% sampled. This will be used in a later version when we predict unobserved catches only, and add them to the observed catches.

Throughout the data specification section, 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 databases 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<-"Example bycatch"

#### Specify name of the observer data file.
obsdat<-read.csv("exampleobs.csv", as.is=TRUE)

#### Specify name of logbook data file. Should be aggregated at stratum level
# or by trip
logdat<-read.csv("examplelog.csv",as.is=TRUE)
```

Next, give the names of the variables in the observer and 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.

```
### 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 s
#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"
logEffort<-"sets"

# Give the name of the column in the logbook data that gives the number of sample units #(trips or sets
logNum<-"trips"  #Usually trips

#Give common and scientific names. Can be a vector of names to do multiple species at the #same time in
#This example takes the columns from a data frame to run multiple species at once.
 common<-c("Grouper, Red" )
 sp<-c("Epinephelus morio")

#Give the name of the columns associated with the species. If it is a vector, it must match #the common
obsCatch<-c("catch.Epinephelus.morio")

#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 intermed
complexModel<-formula(y~(Year+season+EW)^2)
simpleModel<-formula(y~Year)

#The variables must have identical names and factor levels in the observer and logbook #datasets. Speci
#Variables not in this list will retain their original format
factorNames=c("Year","season","EW")
```

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.

```
##Specify which observation error models to try. Options are delta-lognormal, delta-gamma, #negative bi
#using glm.mb in the MASS lbirary, "Tweedie" for cpglm, and TMB nbinom1, nbinom2, and #tweedie in the g
modelTry<-c("Lognormal","Gamma","NegBin","Tweedie","TMBnbinom1","TMBnbinom2","TMBtweedie")

#Specify preferred information criteria for model selection
# Choices are AICc, AIC and BIC. AICc works well.
selectCriteria="AICc"

#Specify whether to run a 10 fold cross-validation (TRUE or FALSE). This may not work with #a small or
DoCrossValidation<-TRUE
```

Preliminary set up and data summaries

From here no changes should be needed to run the code. The next section loads libraries and does some book-keeping, then prints a summary if the input data. This is all run by sourcing a file called "PreliminaryDataSummaries.r".

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 BycatchFunctions.r for all the functions). You will get a message if if any of the variables in the specified models are not found in the data frame.

```
##############################################################################################
### From here on no changes should be needed.  However, there are places where you should #stop and loo

##############################################################################################
#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(tweedie)
library(reshape2)
library(glmmTMB)


#############################################################################
source(paste0(baseDir,"/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 labelled 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, and "cv" for models that produced results with unreasonably high CVs (>10) in the annual catch predictions. 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 for the complete dataset and prints all the output files.
bestmod<-NULL
predbestmod<-list()
allmods<-list()
modelSummary<-matrix(0,numSp,length(modelTry),dimnames=list(common,modelTry))
modelFail<-matrix("-",numSp,length(modelTry)+1,dimnames=list(common,c("Binomial",modelTry)))
rmsetab<-list()
maetab<-list()
outFiles<-list()
residualtab<-list()
run<-1 #Set to 1 for demo.
#for(run in 1:numSp) {
 datval<-dat[[run]]
 residualtab[[run]]<-matrix(0,8,length(modelTry)+1,dimnames=list(c("KS.D","KS.p",
                            "Dispersion.ratio","Dispersion.p" ,
                            "ZeroInf.ratio" ,"ZeroInf.p","Outlier" , "Outlier.p"),
                            c("Binomial",modelTry)))
 outVal<-paste0(dirname[[run]],common[run],catchType[run])
 outFiles[[run]]<-c(fileList[[run]],
   paste0(outVal,rep(c("Fit","Residuals"),length(modelTry)+1),
          rep(dimnames(modelFail)[[2]],each=2),".pdf"),
   paste0(outVal,"AllFit.pdf"))
 if(DoCrossValidation) outFiles[[run]]<-c(outFiles[[run]],                    paste0(ou
```

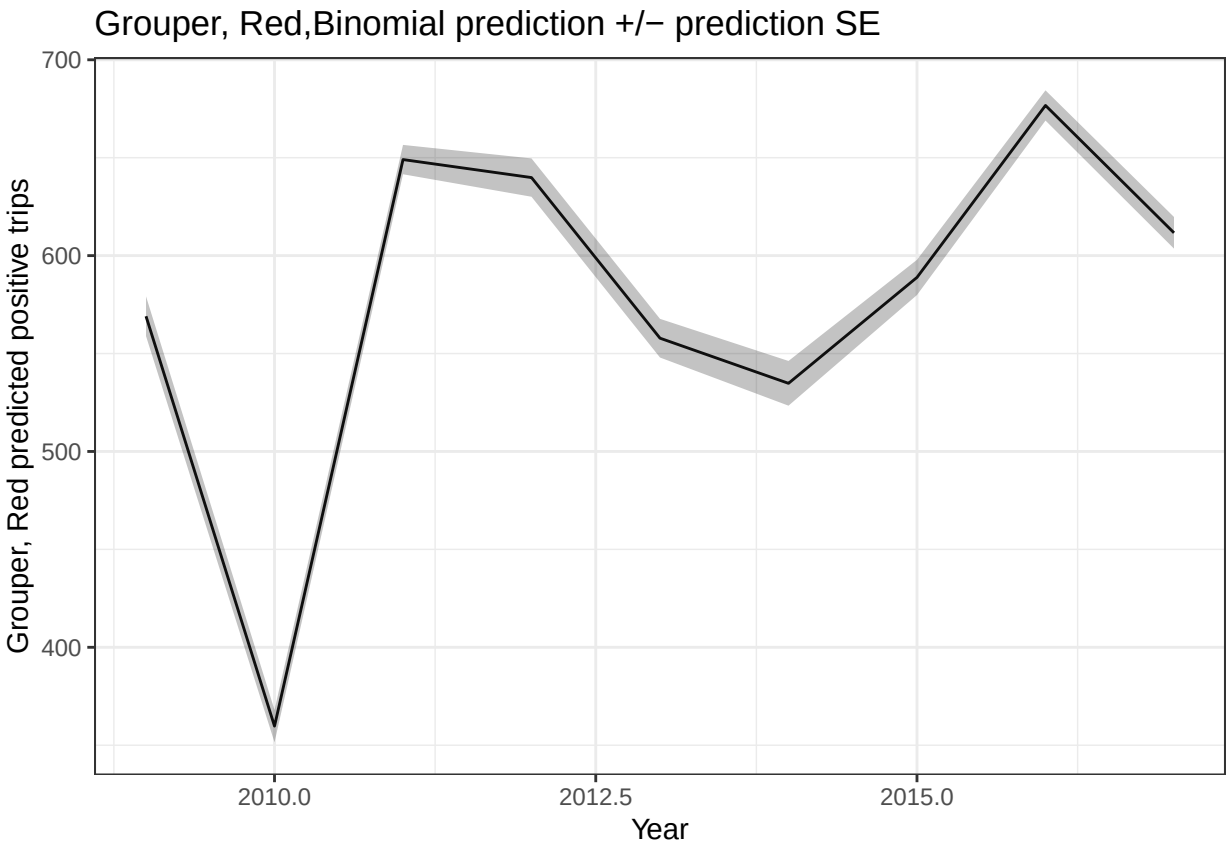
After setting up the data and reprinting the summary, the first model run is the binomial, which is needed for the delta models. The models are fitted using the glm function with a logit link. There is a function called findBestModelFunc, which applies the dredge function from the MuMin library to find 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 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 most 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 residuls 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.

```
 #Find best binomial model and print outputs.
varExclude<-NULL
bin1<-findBestModelFunc(datval,"Binomial",printOutput=TRUE)
if(!is.null(bin1)) {  #If model converged make predictions and print figures
 binpredvals<-makePredictionsVar(bin1,modType="Binomial",newdat=logdat,printOutput=TRUE)
 plotFits(binpredvals,"Binomial",paste0(outVal,"FitBinomial.pdf"))
 residualtab[[run]][,"Binomial"]<-ResidualsFunc(bin1,"Binomial",paste0(outVal,"ResidualsBinomial.pdf"))
 if(is.null(binpredvals)) modelFail[run,"Binomial"]<-"cv"
 } else  {
   modelFail[run,"Binomial"]<-"fit"
   binpredvals<-NULL
 }
```



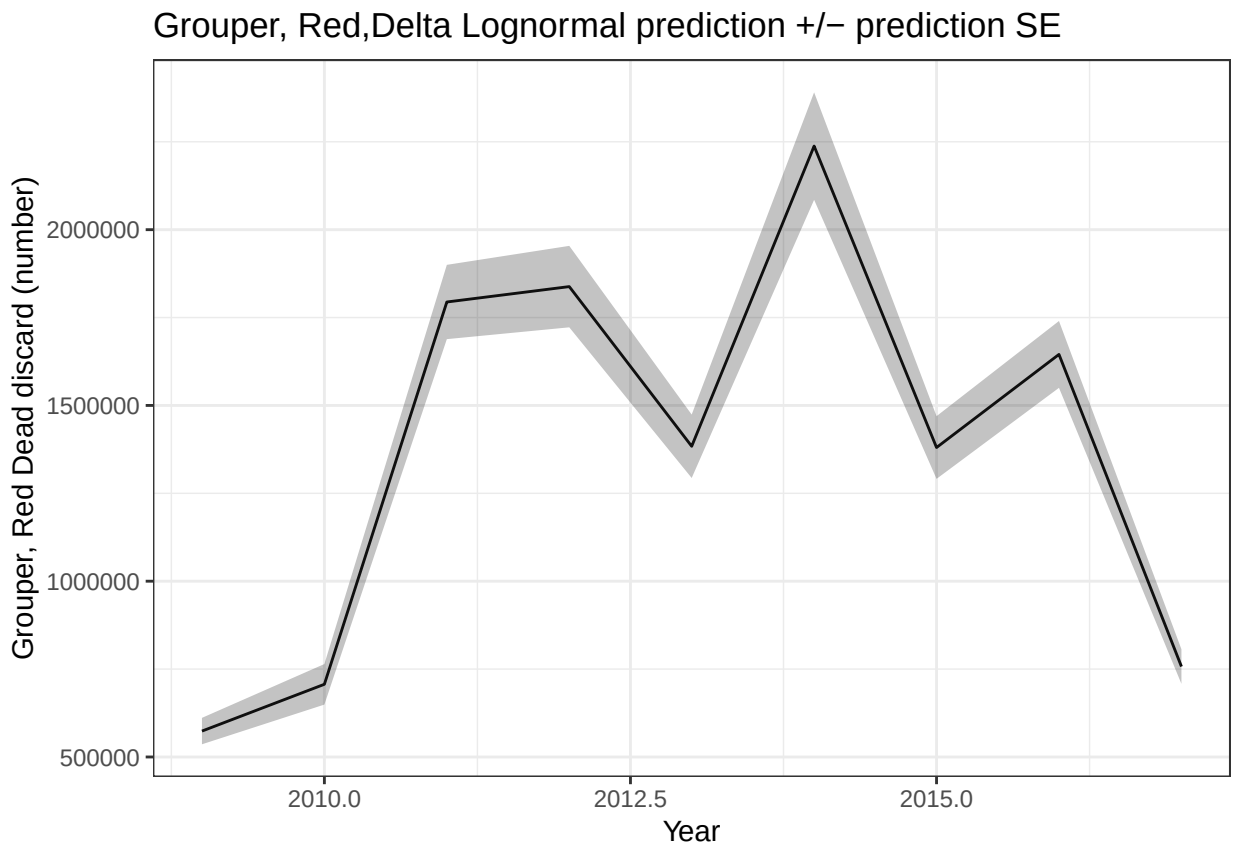
If all years have at least one positive observation, the loop next runs the lognormal and gamma models and calculates total bycatch using the delta-lognormal and delta-gamma methods. For the lognormal model, the CPUE is log transformed, and the mean CPUE for positive observations is modeled with the 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 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 squred times the variance of CPUE. THe total catches and their variances are summed across trips to get the annual totals.

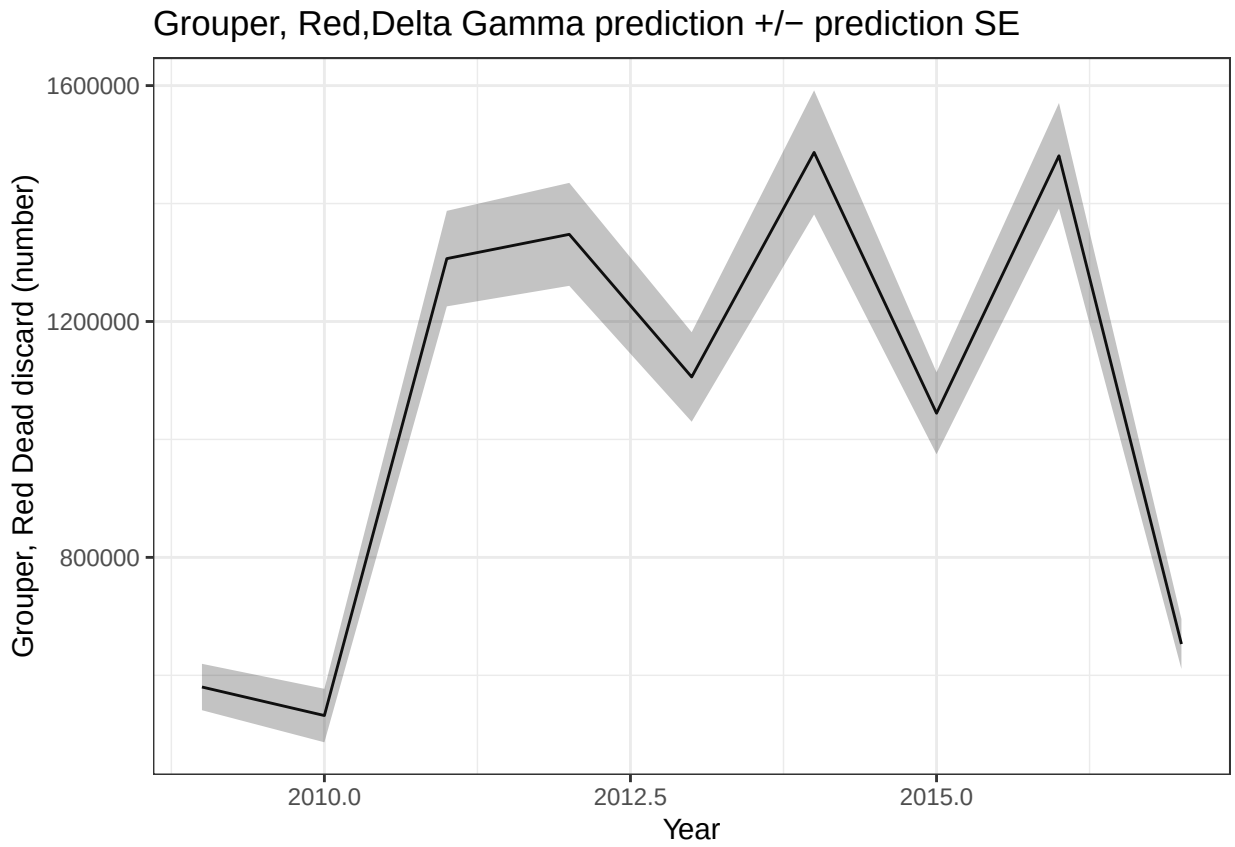
For the gamma method, the default inverse link is used to model the postive CPUE values. The total catch is calculated as the predicted probability of a positive catch times the predicted CPUE times effort. The variance of total catch is calculated as effort squared times the variance of the CPUE calculated 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.

```
predvals=rep(list(NULL),length(modelTry))
names(predvals)=modelTry
modfits=rep(list(NULL),length(modelTry))
```

```
names(modfits)=modelTry
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 are included
  varExclude<-names(y)[y>0]
  if(length(varExclude>0))
    print(paste(common[run], "excluding variable",varExclude,"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
    for(mod in which(modelTry %in% c("Lognormal","Gamma"))) {
      modfits[[modelTry[mod]]]<-findBestModelFunc(posdat,modelTry[mod],printOutput=TRUE)
      if(!is.null(modfits[[modelTry[mod]]])) {
        predvals[[modelTry[mod]]]<-makePredictionsVar(modfit1=bin1,modfit2=modfits[[modelTry[mod]]],mo
        plotFits(predvals[[modelTry[mod]]],modelTry[mod],paste0(outVal,"Fit",modelTry[mod],".pdf"))
        residualtab[[run]][,modelTry[mod]]<-ResidualsFunc(modfits[[modelTry[mod]]],modelTry[mod],paste
        if(is.null(predvals[[modelTry[mod]]])) modelFail[run,modelTry[mod]]<-"cv"
      } else {
        modelFail[run,modelTry[mod]]<-"fit"
      }
    }
  } else {
    print("Not all years have positive observations, skipping delta models")
    modelFail[run,c("Lognormal","Gamma")]<-"data"
  }}
```


Grouper, Red,Delta Lognormal prediction +/− prediction SE
Grouper, Red Dead discard (number)
2000000
1500000
1000000
500000
2010.0
2012.5
2015.0
Year



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 model gives somewhat different results from the others.

The negative binomial predicts integer counts, so it is appropriate for predicting bycatch in numbers per trip. 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:

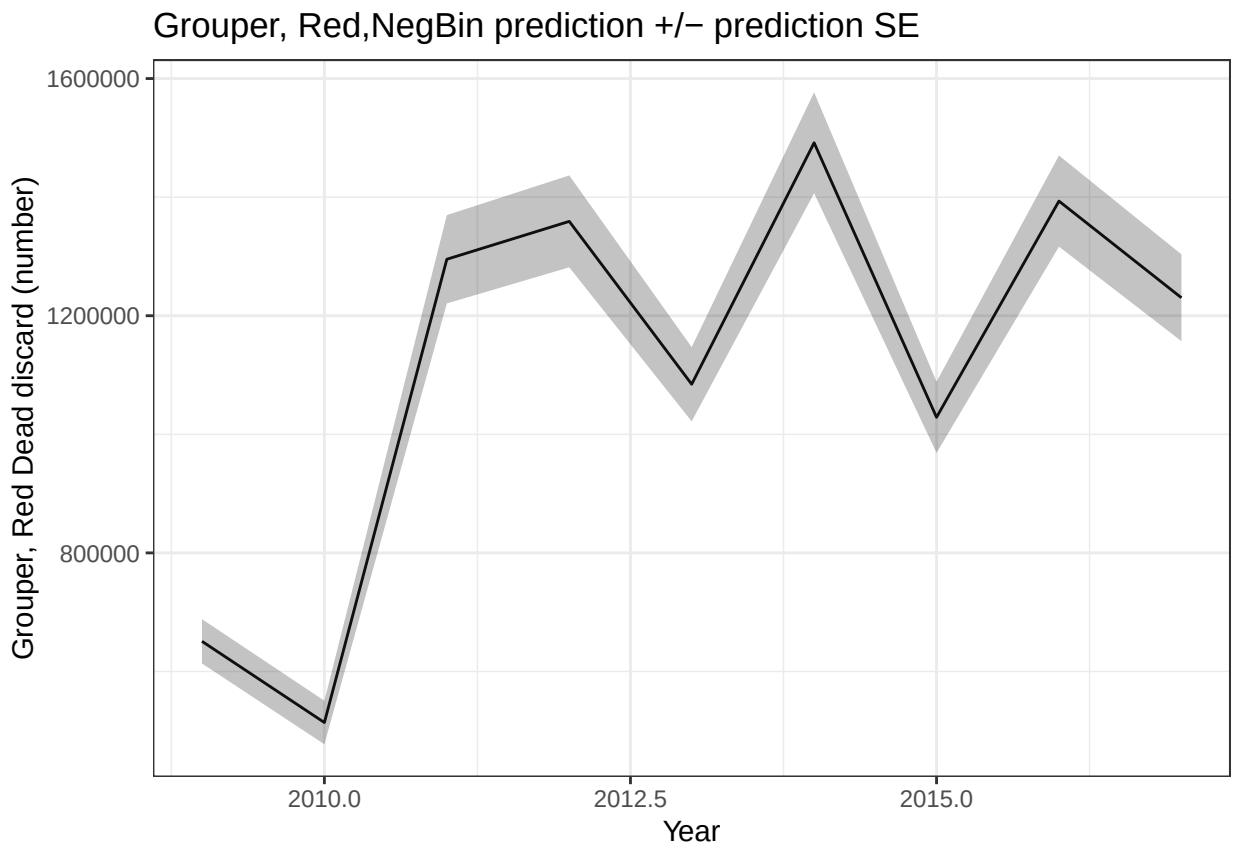
$$log(C_i) = b_0 + b_1 x_1 + offset(log(E_i))$$

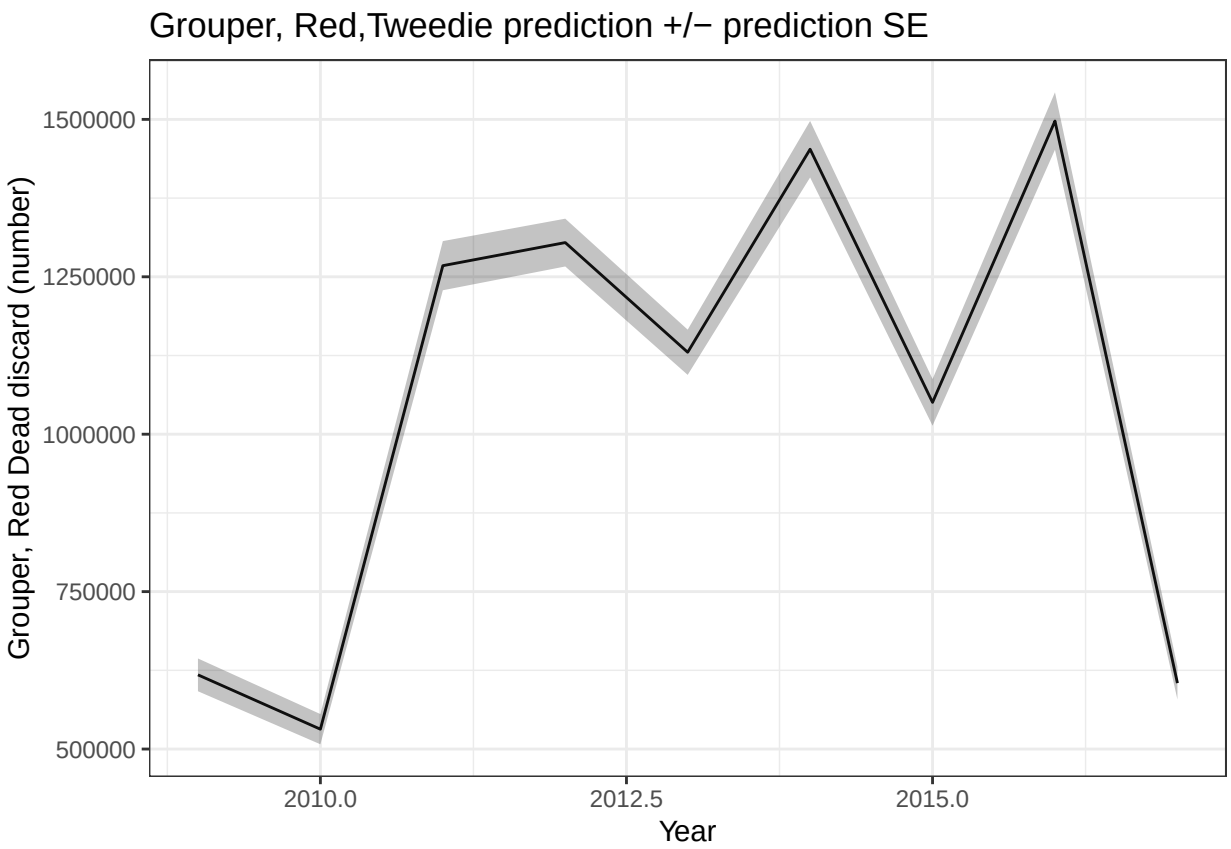
were C_i is the catch in trip i in the observer data, $b_0 + b_1 x_1$ is an example linear predictor with an intercept and a slope, and the offset is the log of the effort E_i in each trip. This is algebraically equivalent to modeling CPUE as a function of the same linear predictor. The total catches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

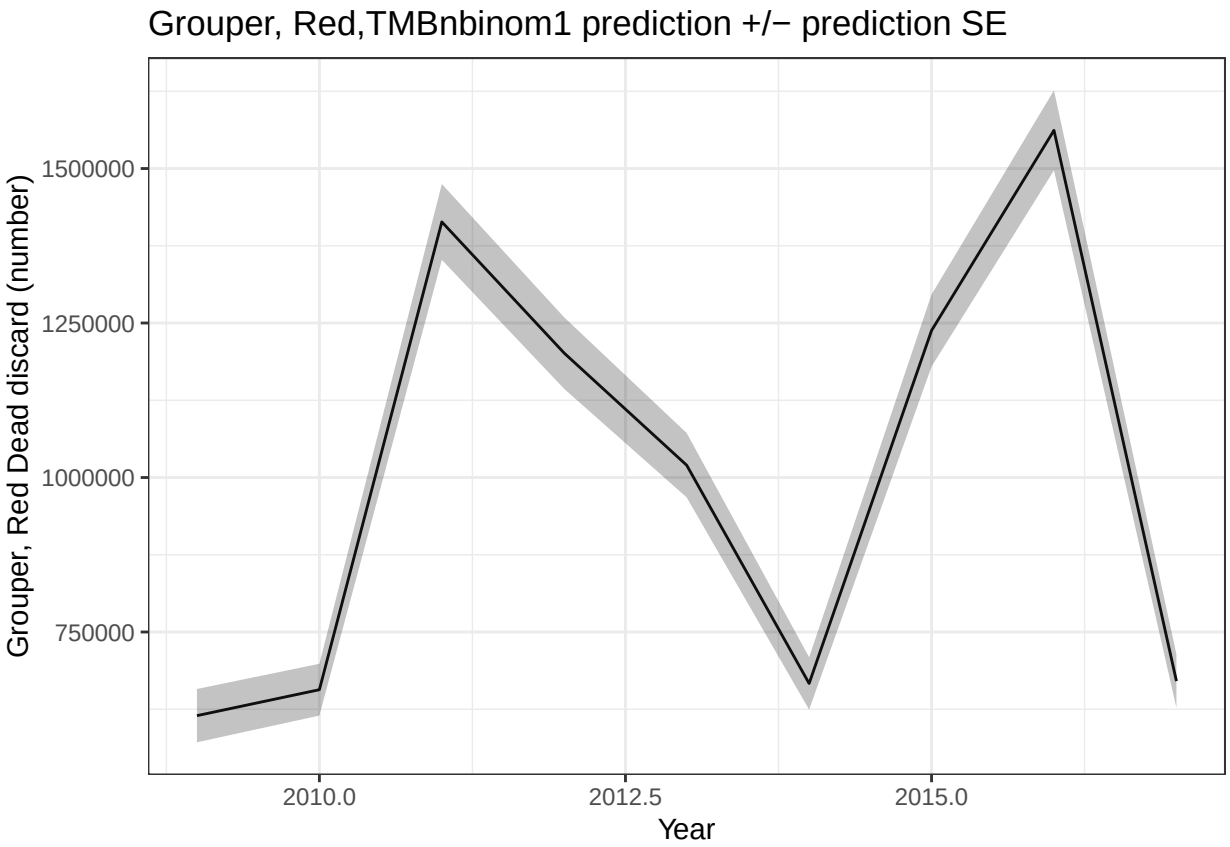
The total catch estimates are plotted, along with the ordinary and DHARMa residuals. For the negative binomial model. Because the response is an integer, the ordinary resiudals tend to have patterns that are difficult to interpret. Thus, the DHARMa residuals are more useful for evaluating model fit for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

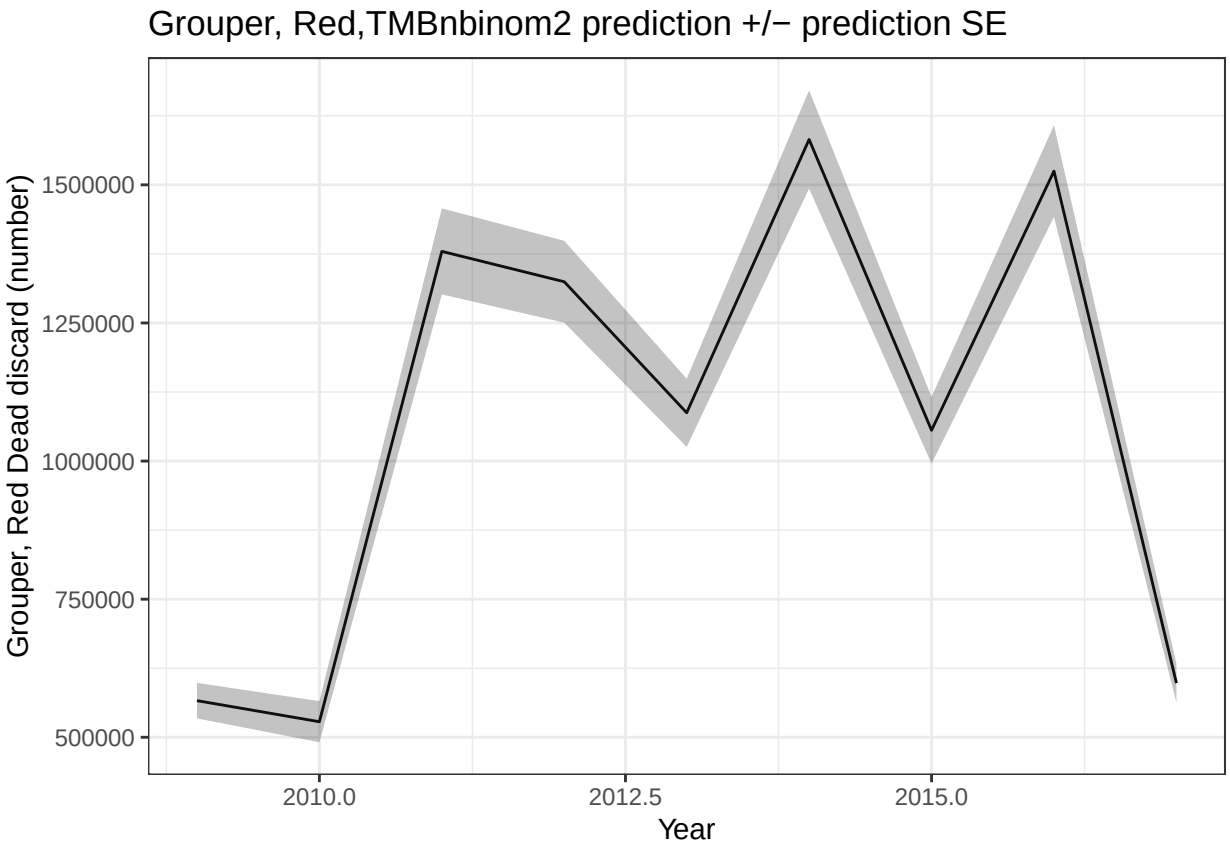
The Tweedie distribution can also be used to model the predicted CPUE, using the cpglm function in the cplm library (Zhang (2013)) or the tweedie family in glmmTMB (Brooks et al. (2017)). The libraries give very similar results. Tweedie is a generalized function that estimates a distribution similar 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 BycatchFunctions.r for details.

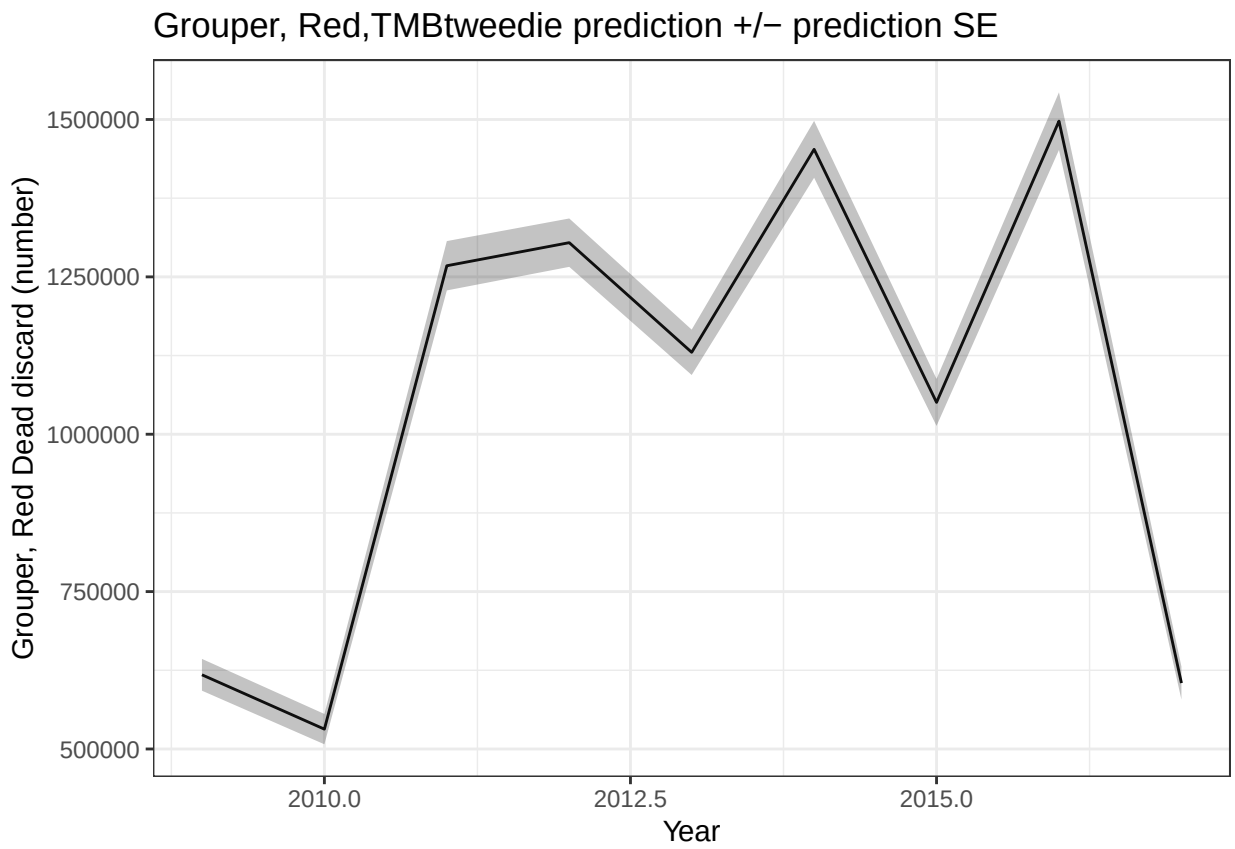
```
#All models except delta
for(mod in which(!modelTry %in% c("Lognormal","Gamma"))) {
  modfits[[modelTry[mod]]]<-findBestModelFunc(datval,modelTry[mod],printOutput=TRUE)
  if(!is.null(modfits[[modelTry[mod]]])) {
    predvals[[modelTry[mod]]]<-makePredictionsVar(modfit1=modfits[[modelTry[mod]]],modType=modelTry[mo
    plotFits(predvals[[modelTry[mod]]],modelTry[mod],paste0(outVal,"Fit",modelTry[mod],".pdf"))
    residualtab[[run]][,modelTry[mod]]<-ResidualsFunc(modfits[[modelTry[mod]]],modelTry[mod],paste0(ou
    if(is.null(predvals[[modelTry[mod]]])) modelFail[run,modelTry[mod]]<-"cv"
  } else {
    modelFail[run,modelTry[mod]]<-"fit"
  }
}
```


Grouper, Red,NegBin prediction +/− prediction SE
1600000
1200000
800000
Grouper, Red Dead discard (number)
2010.0
2012.5
2015.0
Year


Grouper, Red,Tweedie prediction +/− prediction SE
Grouper, Red Dead discard (number)
1500000
1250000
1000000
750000
500000
2010.0
2012.5
2015.0
Year

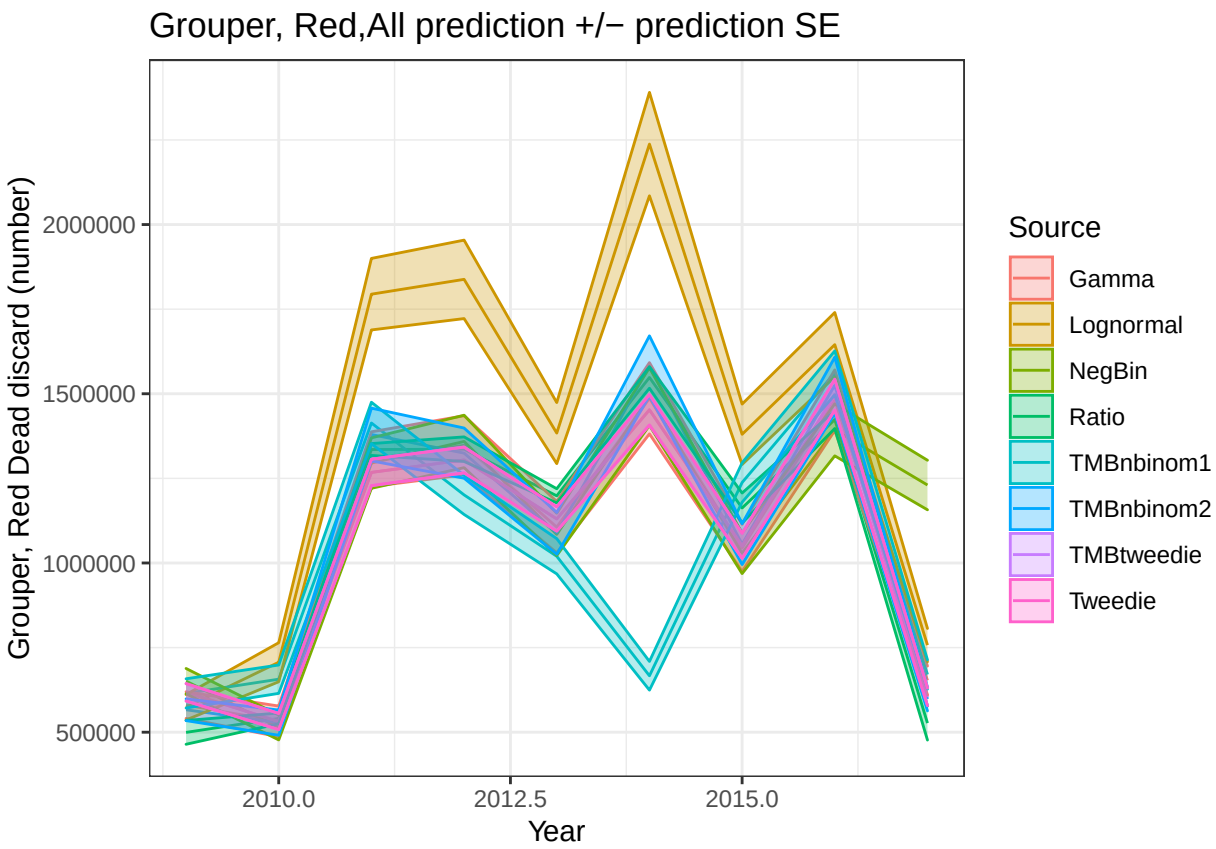

Grouper, Red,TMBnbinom1 prediction +/− prediction SE
Grouper, Red Dead discard (number)
1500000
1250000
1000000
750000
2010.0
2012.5
2015.0
Year


Grouper, Red,TMBnbinom2 prediction +/− prediction SE
Grouper, Red Dead discard (number)
1500000
1250000
1000000
750000
500000
2010.0
2012.5
2015.0
Year



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 that 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-disperions, 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
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(predvals,list(Ratio=yearsumgraph)),.id="Source")
plotFits(allmods[[run]],modType="All",paste0(outVal,"AllFit.pdf"))
```




```
#Show the diagnostic table
printTableFunc("Diagnostics",sp[run],residualtab[[run]],paste0(outVal,"residualDiagnostics.pdf"),useR
```

Diagnostics

Epinephelus morio

	Binomial	Lognormal	Gamma	NegBin	Tweedie	TMBnbinom1	TMBnbinom2	TMBtweedie
KS.D	0.03	0.16	0.13	0.21	0.08	0.18	0.23	0.08
KS.p	0.82	0.00	0.00	0.00	0.02	0.00	0.00	0.01
Dispersion.ratio	1.01	0.98	0.78	0.41	0.86	0.58	0.42	0.85
Dispersion.p	0.94	0.66	0.01	0.00	0.00	0.00	0.00	0.00
ZeroInf.ratio	1.01	NaN	NaN	2.09	1.16	1.31	1.90	1.15
ZeroInf.p	0.97	1.00	1.00	0.00	0.08	0.00	0.00	0.08
Outlier	0.00	8.00	6.00	0.00	3.00	4.00	1.00	3.00
Outlier.p	1.00	0.04	0.44	0.06	0.88	0.86	0.56	0.86

```
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:

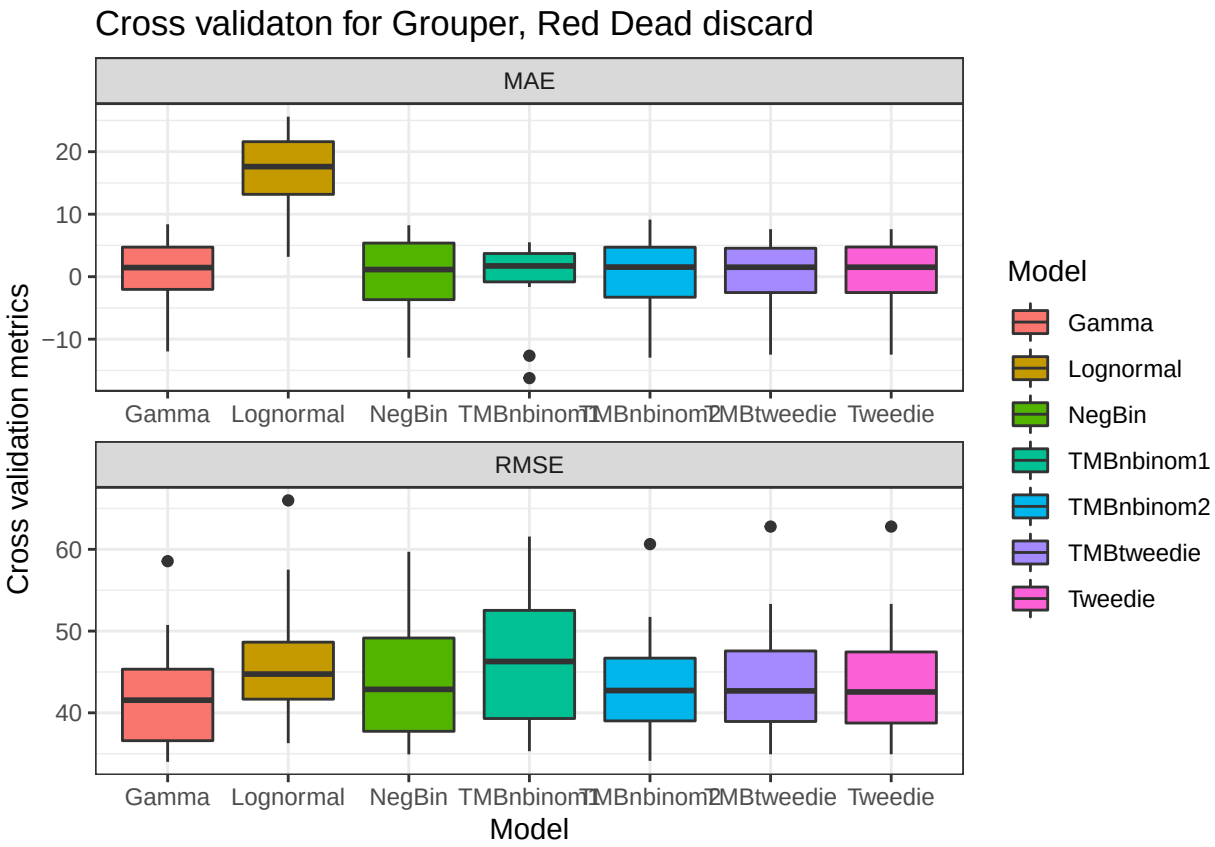
$$RMSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

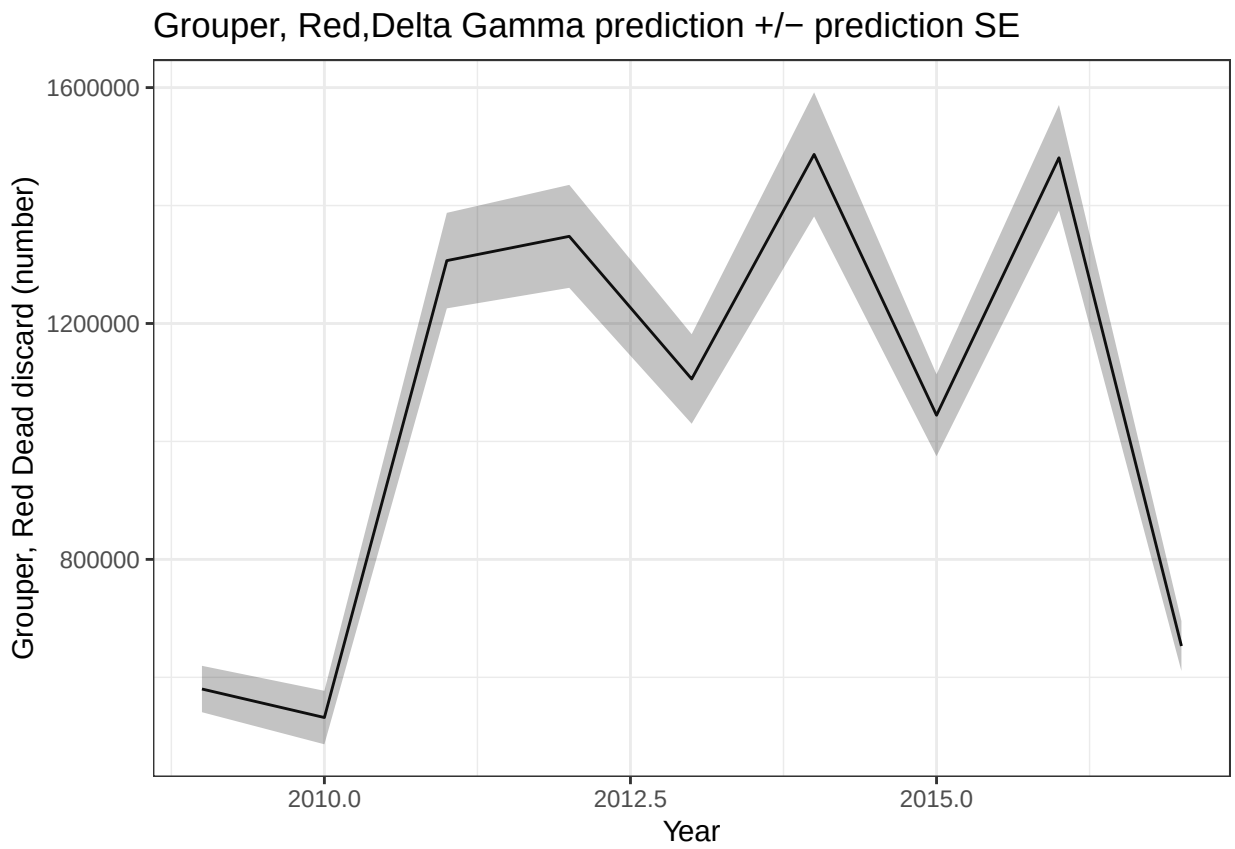
where n is the number of observed trips and y is the CPUE data in the left-out tenth of the observer data, and $\hat{y}$ is the CPUE predicted from the model fitted to the other 9/10th of the observer data. The model with the lowest mean RMSE across the 10 folds is selected as the best model. Mean average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

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

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
  datval$cvsample<-sample(rep(1:10,length=dim(datval)[1]),replace=FALSE)
  table(datval$cvsample,datval$Year)
  rmsetab[[run]]<-data.frame(matrix(NA,10,length(modelTry),dimnames=list(1:10,modelTry)))
  maetab[[run]]<-rmsetab[[run]]
  for(i in 1:10 ) {
   datin<-datval[datval$cvsample!=i,]
   datout<-datval[datval$cvsample==i,]
   datout$SampleUnits<-rep(1,dim(datout)[1])
   bin1<-findBestModelFunc(datin,"Binomial")
   if("Lognormal" %in% modelTry | "Gamma" %in% modelTry) {
     posdat<-filter(dat[[run]],pres==1)
     for(mod in which(modelTry %in% c("Lognormal","Gamma"))) {
       if(modelFail[run,modelTry[mod]]=="-" & min(summary(posdat$Year))>0) {
         modfit1<-findBestModelFunc(posdat,modelTry[mod])
         predcpue<-makePredictions(bin1,modfit1,modelTry[mod],datout)
         rmsetab[[run]][i,modelTry[mod]]<-getRMSE(predcpue$est.cpue,datout$cpue)
         maetab[[run]][i,modelTry[mod]]<-getMAE(predcpue$est.cpue,datout$cpue)
       }
     }
   }
   for(mod in which(!modelTry %in% c("Lognormal","Gamma"))) {
     if(modelFail[run,modelTry[mod]]=="-") {
       modfit1<-findBestModelFunc(datin,modelTry[mod])
       predcpue<-makePredictions(modfit1,modType=modelTry[mod], newdat = datout)
       rmsetab[[run]][i,modelTry[mod]]<-getRMSE(predcpue$est.cpue,datout$cpue)
       maetab[[run]][i,modelTry[mod]]<-getMAE(predcpue$est.cpue,datout$cpue)
     }
   }
  }
 write.csv(rmsetab[[run]],paste0(outVal,"rmse.csv"))
 write.csv(maetab[[run]],paste0(outVal,"mae.csv"))
 plotCrossVal(rmsetab[[run]],maetab[[run]],paste0(outVal,"CrossValidation.pdf"))
 #Select best model based on cross validation
 modelSummary[run,]<-apply(rmsetab[[run]],2,mean,na.rm=TRUE)
 best<-which(!is.na(modelSummary[run,]) &modelSummary[run,]==min(modelSummary[run,],na.rm=TRUE))
 bestmod[run]<-dimnames(modelSummary)[[2]][best]
 predbestmod[[run]]<-predvals[[modelTry[best]]]
 plotFits(predbestmod[[run]],names(best),paste0(outVal,"BestFit.pdf"))
}
```


Cross validaton for Grouper, Red Dead discard
MAE
RMSE
Cross validation metrics
20
10
0
−10
60
50
40
Gamma
Lognormal
NegBin
TMBnbinom1
TMBnbinom2
TMBtweedie
Tweedie
Model
Model
Gamma
Lognormal
NegBin
TMBnbinom1
TMBnbinom2
TMBtweedie
Tweedie




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

```
## [1] "Grouper, Red complete"
```

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

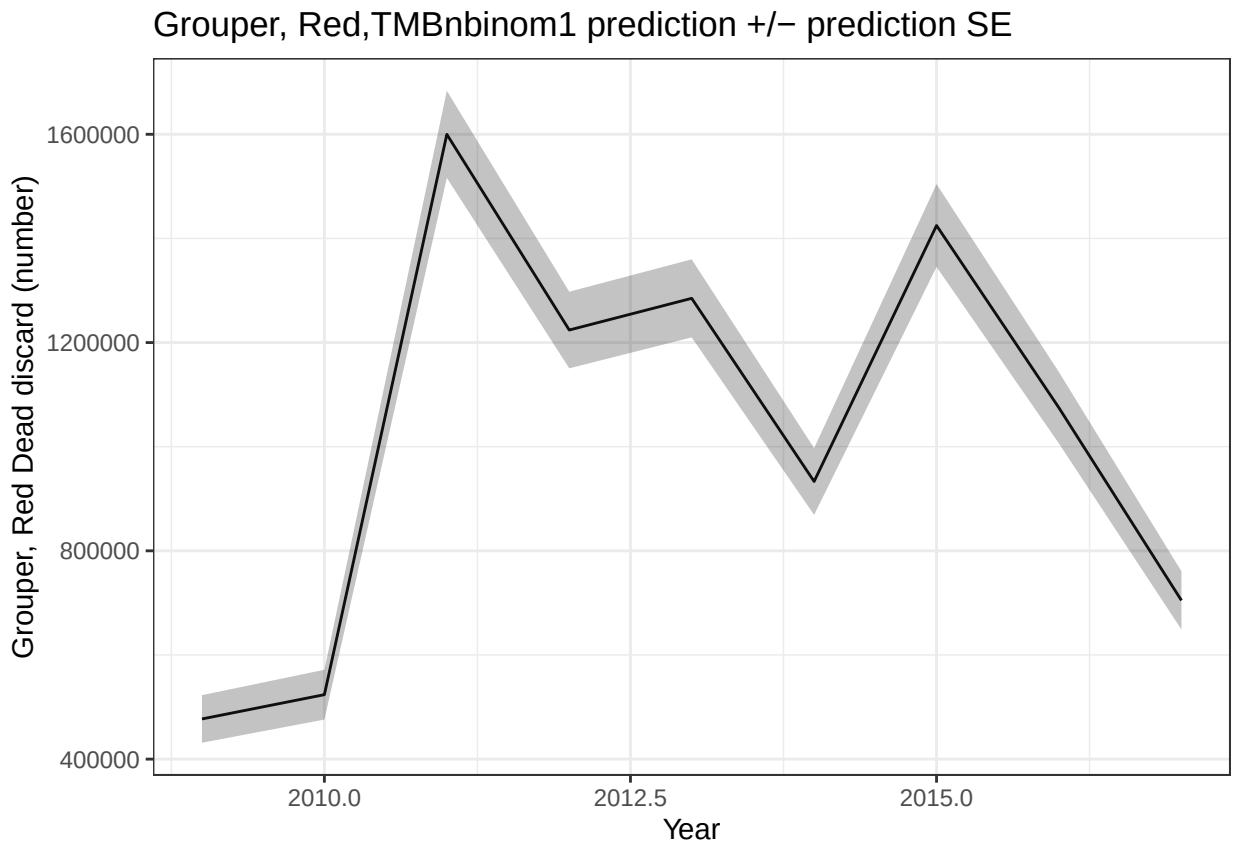
At the end of the analysis loop, summaries such as the table of whether each model configuration failed, the best fit annual summaries, and the crossvalidation results are printed to the main output file.

The next section allows the user to specify a particular model, and print out all its summary statistics and diagnostics. This is useful if you want to chose a model other than the RMSE best model and look at it more closely.

```
## Select a particular model and look at its outputs. You must specify the
# observation error model and formulas for both components of delta models. For non-delta  # models, le
# Indicate the number of the species from the list of species run above.
## This works after running the main loop.
#Options 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",paste0(selectOutDir,"
 ResidualsFunc(modlist[[1]],ModelSelect,paste0(selectOutDir,"/",ModelSelect,"Residuals.pdf"))
```

```
##             KS.D             KS.p Dispersion.ratio     Dispersion.p
##        0.2443077        0.0000000        0.4993779        0.0000000
##    ZeroInf.ratio        ZeroInf.p          Outlier        Outlier.p
##        1.4944184        0.0000000        4.0000000        0.7000000
```

```
if(!is.null(modlist[[2]])) ResidualsFunc(modlist[[2]],ModelSelect,paste0(selectOutDir,"/",ModelSelect,"
```

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 residuals and 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.

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.

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.

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