

Supplementary Material: Estimation of Time-Varying Natural Mortality and Reporting Rates in a multi-region Integrated Tagging and Catch-at-Age Analysis

This document contains additional information that was not included in the main manuscript. Additional boxplots of relative error and actual error for parameters not presented in the manuscript are presented. Also an explanation of the results for some of the parameters presented in the Supplementary Materials is given.

Methods

Performance Metrics

The calculation of the relative error for a few of the estimated parameters contain some caveats. Within the ITCAAN model the estimation of the selectivity at age was allowed to have a value greater than 1. Therefore, additional steps were conducted to calculate the relative errors for the selectivity parameters and catchability coefficients. All selectivity-at-age parameters were divided by the maximum estimated selectivity-at-age parameter so that the new maximum value would equal 1. The relative errors for each age, individually, were then calculated using these adjusted values and all REEs were plotted in the boxplots. The REE for the catchability coefficients also needed an additional step since the catchability coefficients are multiplied by the selectivity-at-age parameters to estimate the fishing mortality within the ITCAAN model. If the maximum selectivity-at-age parameter within the ITCAAN model were estimated to be larger than 1 then the catchability coefficient would be underestimated to compensate. To correct for this fact the annual catchability coefficient estimates were multiplied by the maximum estimated selectivity-at-age before calculating the REEs. The annual estimates of catchability coefficients across years were then combined into the single boxplot shown for each scenario.

Movement rate estimates for the ITCAAN model were separated into two different groups to show a more precise picture of the estimates. The first group of REEs calculated was for the proportion of the population that stayed within the natal region (the largest proportion of the population) and is referred to as the Stay Rate in the figures. The second group of movement REEs is the proportion of the population that moved to all other non-natal regions (the number after which the scenario is named) and is referred to as the Move Rate in the figures. The Actual Errors were also calculated for these groups of movement parameters because the small movement rate at the 1% scenario resulted in very large REEs as a result of the very small movement proportion for the Move Rate but the estimates have very small IQR on the actual error scale. Both results are presented here to allow readers to come to their own conclusions.

Table 1: Fishery specific parameters for investigated scenarios (abbreviations in Table 1, where Reg is region, Comm is commercial fishery, Rec is recreational fishery, and AR1 is first order autoregressive process).

| Scenario | Reg 1 Comm | Reg 1 Rec | Reg 2 Comm | Reg 2 Rec | Reg 3 Comm | Reg 3 Rec | Reg 4 Comm | Reg 4 Rec |
|---|---------------|--------------|---------------|--------------|---------------|--------------|---------------|--------------|
| Constant Reporting Rate: NoRwrNoLoss, NoRwrLoss, RwrLoss, and TVM | 0.25 | 0.35 | 0.15 | 0.40 | 0.20 | 0.30 | 0.10 | 0.45 |
| Average Reporting Rate: TVRR and BothTV | 0.15 | 0.43 | 0.15 | 0.43 | 0.15 | 0.43 | 0.15 | 0.43 |
| Reporting rate in first year: DecrRR, Rwr5, Rwr10, and Season | 0.25 | 0.50 | 0.20 | 0.55 | 0.25 | 0.50 | 0.20 | 0.60 |
| Reporting rate in last year: DecrRR, Rwr5, Rwr10, and Season | 0.05 | 0.25 | 0.10 | 0.20 | 0.10 | 0.15 | 0.05 | 0.20 |
| Mean apical F: All scenarios | 0.2773 | 0.0885 | 0.0885 | 0.0628 | 0.1556 | 0.1232 | 0.1232 | 0.1690 |
| Catchability coefficients: All scenarios | 2e-6 | 2e-8 | 5e-6 | 1e-5 | 6e-7 | 3e-5 | 8e-7 | 3e-7 |

Table 2: Parameters in the data generating model that are unique for each age.

| Parameter | Age 2 | Age 3 | Age 4 | Age 5 | Age 6 | Age 7 |
|---|--------|--------|--------|--------|-------|--------|
| Weight | 0.8347 | 1.1659 | 1.4875 | 1.7687 | 1.944 | 2.3323 |
| Maturity | 0.308 | 0.824 | 0.914 | 0.935 | 0.978 | 1 |
| Survey selectivity region 1 | 0.6 | 0.7 | 1 | 0.9 | 0.9 | 0.9 |
| Survey selectivity region 2 | 1 | 0.8 | 0.6 | 0.55 | 0.55 | 0.3 |
| Survey selectivity region 3 | 1 | 0.5 | 0.4 | 0.3 | 0.3 | 0.3 |
| Survey selectivity region 4 | 1 | 1 | 1 | 1 | 1 | 1 |
| Fishery selectivity region 1 commercial | 0.35 | 0.98 | 1 | 0.7 | 0.5 | 0.5 |
| Fishery selectivity region 1 recreational | 0.04 | 0.5 | 0.75 | .85 | .92 | 1 |
| Fishery selectivity region 2 commercial | 0.4 | 1 | 0.9 | 0.8 | 0.8 | 0.7 |
| Fishery selectivity region 2 recreational | 0.5 | 0.9 | 1 | 0.7 | 0.65 | 0.6 |
| Fishery selectivity region 3 commercial | 0.4 | 1 | 0.9 | 0.8 | 0.8 | 0.7 |
| Fishery selectivity region 3 recreational | 0.1 | 0.6 | 0.65 | 0.7 | 0.8 | 1 |
| Fishery selectivity region 4 commercial | 0.01 | 0.13 | 0.35 | 1 | 1 | 1 |
| Fishery selectivity region 4 recreational | 0.01 | 0.15 | 1 | 1 | 1 | 1 |
| Allocation of tags | 5% | 10% | 20% | 20% | 20% | 25% |

Table 3: Parameters used to simulate data for each of the four regions in the simulation model.

| Parameter | Region 1 | Region 2 | Region 3 | Region 4 |
|--|--|-----------|----------|------------|
| Ricker stock recruit α | 2.41807 | 1.48449 | 1 | 0.34915 |
| Ricker stock recruit β | 1.29135e-6 | 3.0618e-8 | 10e-6 | 2.80287e-8 |
| Survey catchability | 1.5e-5 | 5e-6 | 2e-7 | 8e-7 |
| Catch CV | 0.1 | 0.1 | 0.1 | 0.1 |
| Effort CV | 0.1 | 0.1 | 0.1 | 0.1 |
| Survey CV | 0.2 | 0.2 | 0.2 | 0.2 |
| Initial abundance CV | 0.3 | 0.3 | 0.3 | 0.3 |
| Data generating harvest age comp. samp size | 100 | 100 | 100 | 100 |
| Estimating model harvest age composition ESS | 100 | 100 | 100 | 100 |
| Data generating survey age comp. samp size | 100 | 100 | 100 | 100 |
| Estimating model survey age composition ESS | 100 | 100 | 100 | 100 |
| ρ_s | $\sim \text{Trunc. Normal}(0.466, 0.260, -0.99, 0.99)$ | | | |
| σ_s | $\sim \text{Trunc. Normal}(0.777, 0.313, 0, \infty)$ | | | |

Table 4: Tag retention probability for years after release used to simulate tag dynamics for each of the four regions in the simulation model.

| Years After Release | Region 1 | Region 2 | Region 3 | Region 4 |
|---------------------|----------|----------|----------|----------|
| 1 | 0.9139 | 0.7180 | 0.717994 | 0.92306 |
| 2 | 0.8436 | 0.5426 | 0.542648 | 0.86066 |
| 3 | 0.7788 | 0.4101 | 0.410124 | 0.80247 |
| 4 | 0.7189 | 0.3100 | 0.309965 | 0.74822 |
| 5 | 0.6636 | 0.2344 | 0.234267 | 0.69764 |
| 6 | 0.6126 | 0.1771 | 0.177055 | 0.65047 |
| 7 | 0.5655 | 0.1338 | 0.133815 | 0.60650 |
| 8 | 0.5220 | 0.1011 | 0.101135 | 0.56549 |
| 9 | 0.4819 | 0.0764 | 0.076436 | 0.52726 |
| 10 | 0.4448 | 0.0578 | 0.057769 | 0.49161 |
| 11 | 0.4106 | 0.0437 | 0.043661 | 0.45838 |
| 12 | 0.3791 | 0.0330 | 0.032998 | 0.42739 |
| 13 | 0.3499 | 0.0249 | 0.024939 | 0.39849 |
| 14 | 0.3230 | 0.0188 | 0.018849 | 0.37155 |
| 15 | 0.2982 | 0.0142 | 0.014245 | 0.34643 |
| 16 | 0.2753 | 0.0108 | 0.010766 | 0.32301 |
| 17 | 0.2541 | 0.0081 | 0.008137 | 0.30117 |
| 18 | 0.2346 | 0.0062 | 0.006150 | 0.28081 |
| 19 | 0.2165 | 0.0046 | 0.004648 | 0.26183 |
| 20 | 0.1999 | 0.0035 | 0.003512 | 0.24413 |
| 21 | 0.1845 | 0.0027 | 0.002655 | 0.22762 |
| 22 | 0.1703 | 0.0020 | 0.002006 | 0.21223 |
| 23 | 0.1572 | 0.0015 | 0.001516 | 0.19788 |
| 24 | 0.1451 | 0.0011 | 0.001146 | 0.18451 |
| 25 | 0.1340 | 0.0009 | 0.000866 | 0.17203 |
| 26 | 0.1237 | 0.0007 | 0.000654 | 0.16040 |
| 27 | 0.1142 | 0.0005 | 0.000494 | 0.14956 |
| 28 | 0.1054 | 0.0004 | 0.000373 | 0.13944 |
| 29 | 0.0973 | 0.0003 | 0.000282 | 0.13002 |
| 30 | 0.0898 | 0.0002 | 0.000213 | 0.12123 |
| 31 | 0.0829 | 0.0002 | 0.000161 | 0.11303 |
| 32 | 0.0765 | 0.0001 | 0.000122 | 0.10539 |
| 33 | 0.0706 | 9.22e-05 | 9.22e-05 | 0.09826 |
| 34 | 0.0652 | 6.97e-05 | 6.97e-05 | 0.09162 |
| 35 | 0.0602 | 5.27e-05 | 5.27e-05 | 0.08543 |
| 36 | 0.0556 | 3.98e-05 | 3.98e-05 | 0.07965 |
| 37 | 0.0513 | 3.01e-05 | 3.01e-05 | 0.07426 |
| 38 | 0.0474 | 2.27e-05 | 2.27e-05 | 0.06924 |
| 39 | 0.0437 | 1.72e-05 | 1.72e-05 | 0.06456 |
| 40 | 0.0403 | 1.30e-05 | 1.30e-05 | 0.06020 |

Results

Fishery and Survey Selectivity-at-Age and Catchability Coefficient

Most of the ITCAAN estimates of selectivity-at-age for the surveys and the fisheries were unbiased and the different scenarios tested had little impact on the bias or precision of estimates, except for RV1Cancel. However, for some regions the estimates of selectivity-at-age appear to be negatively biased. This is an artifact of the true parameter values set within the operating model. The survey selectivity REEs for region 4 are all equal to or below 0. This result is because all selectivity-at-age parameters for the survey in this region were set equal to 1 and the presented results are the relative error of the estimated selectivity parameters divided by the maximum estimated parameter for that simulation. Therefore, it is not possible to have a relative error that is larger than 0 for the survey selectivity-at-age parameters in region 4. Likewise, there is a slight negative bias in the fishery selectivity for region 4. This is a result of half of the selectivity-at-age parameters (ages 5 – 7) for this fishery being equal to one. Therefore, half of the parameters cannot have an REE value greater than 0 based on the procedures used to calculate the REE and thus this parameter appears to be underestimated. This resulted in the slightly negative bias in the median estimates of selectivity-at-age for this fishery. These negative biases in selectivity-at-age estimates translate to a slight overestimation of the catchability coefficients for the fishery and the survey in region 4. This bias for region 4 is consistent between Group 1 scenarios, but the IQR is larger for scenarios where natural mortality is estimated. However, the catchability coefficient parameters for all regions had significant bias when natural mortality was incorrectly specified in the ITCAAN model (RSensO and RSensU). This bias in catchability coefficient parameters was most pronounced when natural mortality was specified at 0.5 times the true value in the ITCAAN model. Incorrectly specifying the reporting rate did not have much influence on the catchability coefficient estimates for the surveys and fisheries. Likewise, the estimates of selectivity-at-age for the fisheries and surveys were biased when the natural mortality was incorrectly specified in the ITCAAN model. However, the selectivity-at-age estimates for the fisheries and the surveys did not seem to be as strongly affected by incorrectly specifying the reporting rate. Bias in catchability coefficients and selectivity-at-age for some regions was present when reporting rates were spatially variable but assumed constant in the ITCAAN model (RV1Canc and RV2Canc). However, when reporting rate was spatially variable and estimated there was not much influence on selectivity-at-age and catchability coefficient parameter estimates.

Additional Figures

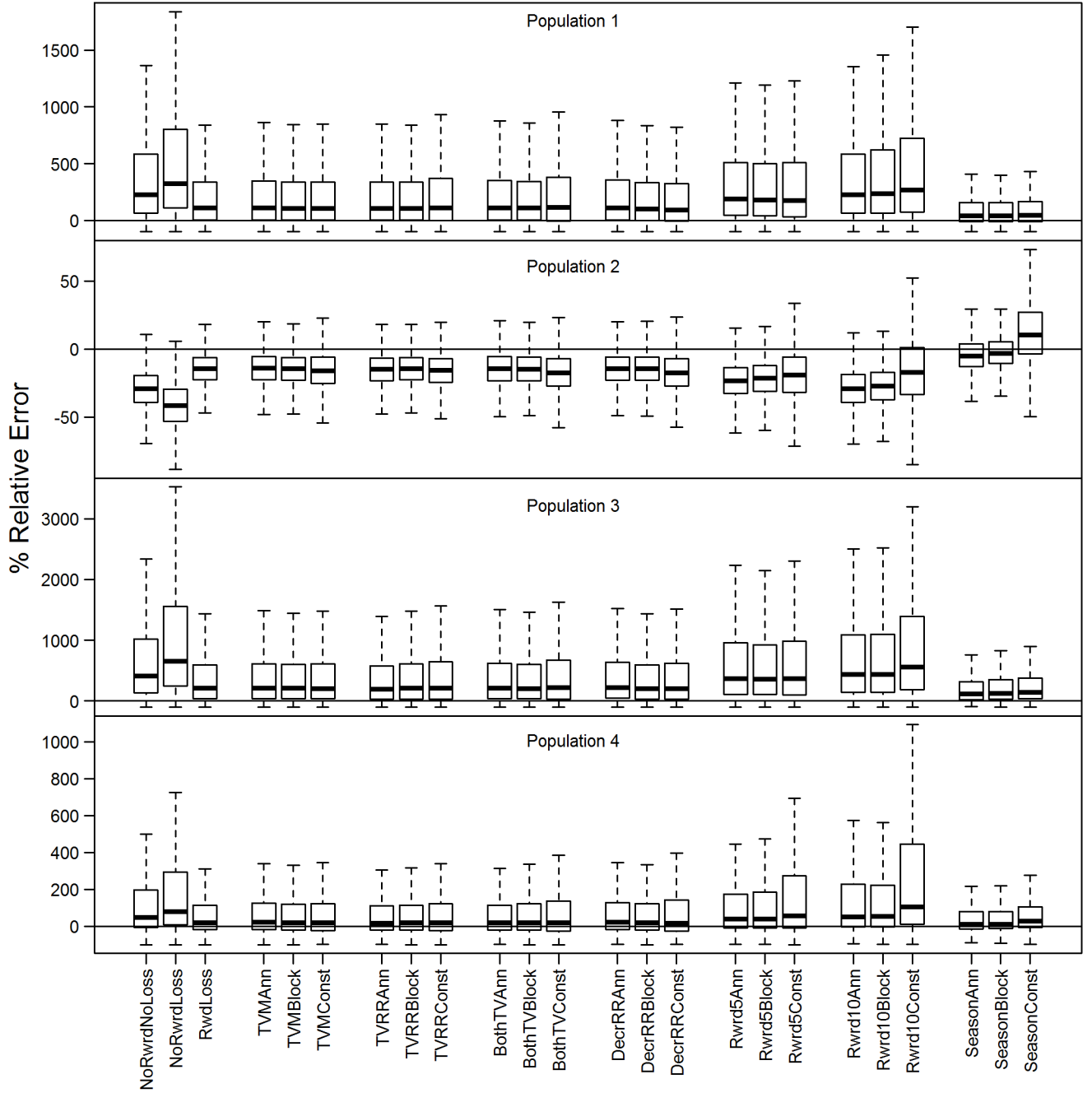


Figure 1: Relative error (%) of recruitment for a four region ITCAAN model under differing assumptions of tag loss, frequency of high reward tag releases, whether movement was simulated annually or seasonally, the true dynamics of natural mortality and reporting rates, and parameterization in the ITCAAN model. Note the difference in scale between the populations. Table 2 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the interquartile range or the most extreme relative error, whichever is less extreme. Relative errors outside the whisker range were not plotted.

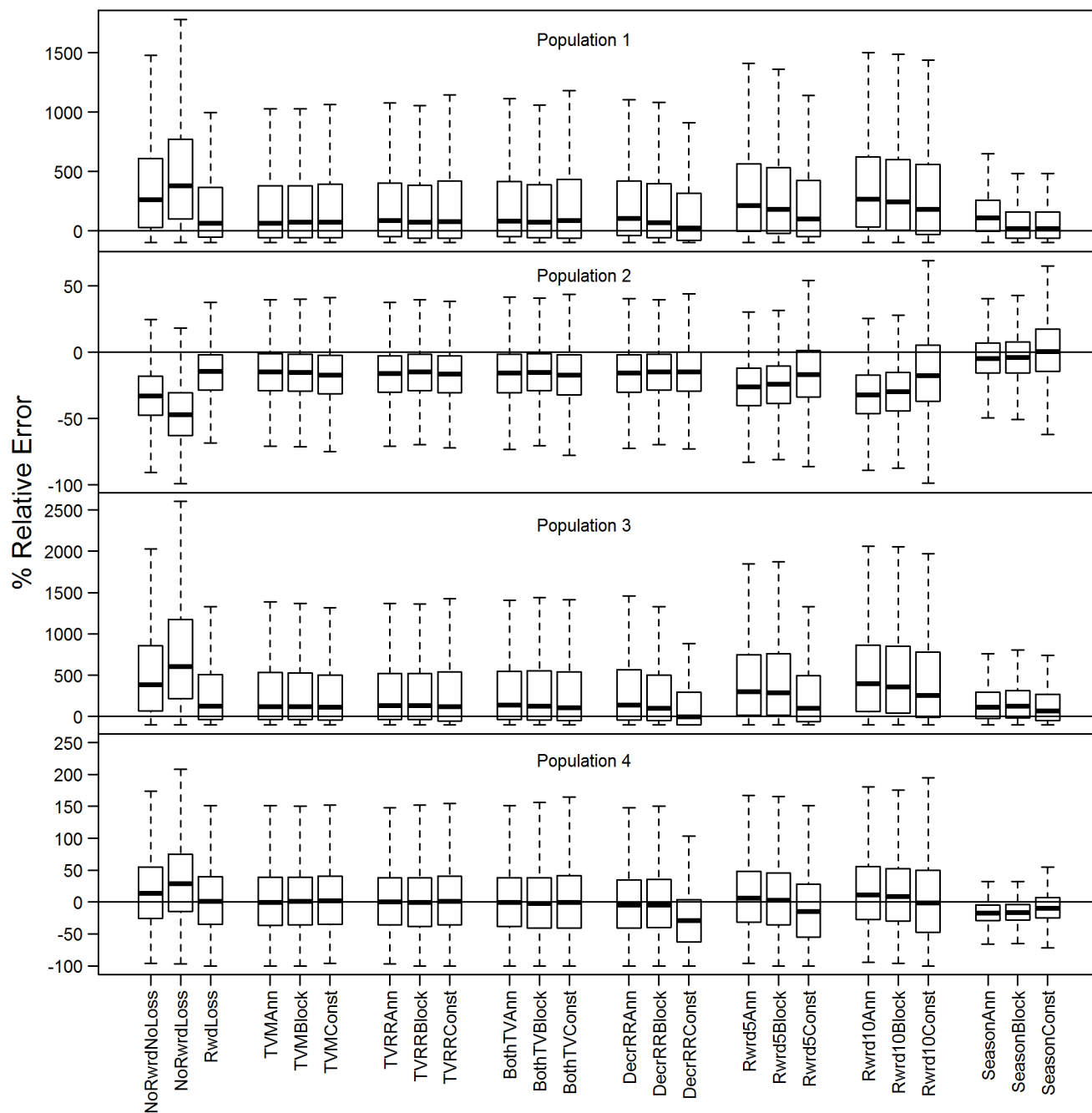


Figure 2: Same as Figure 1 except for abundance in the first year of the model for each population.

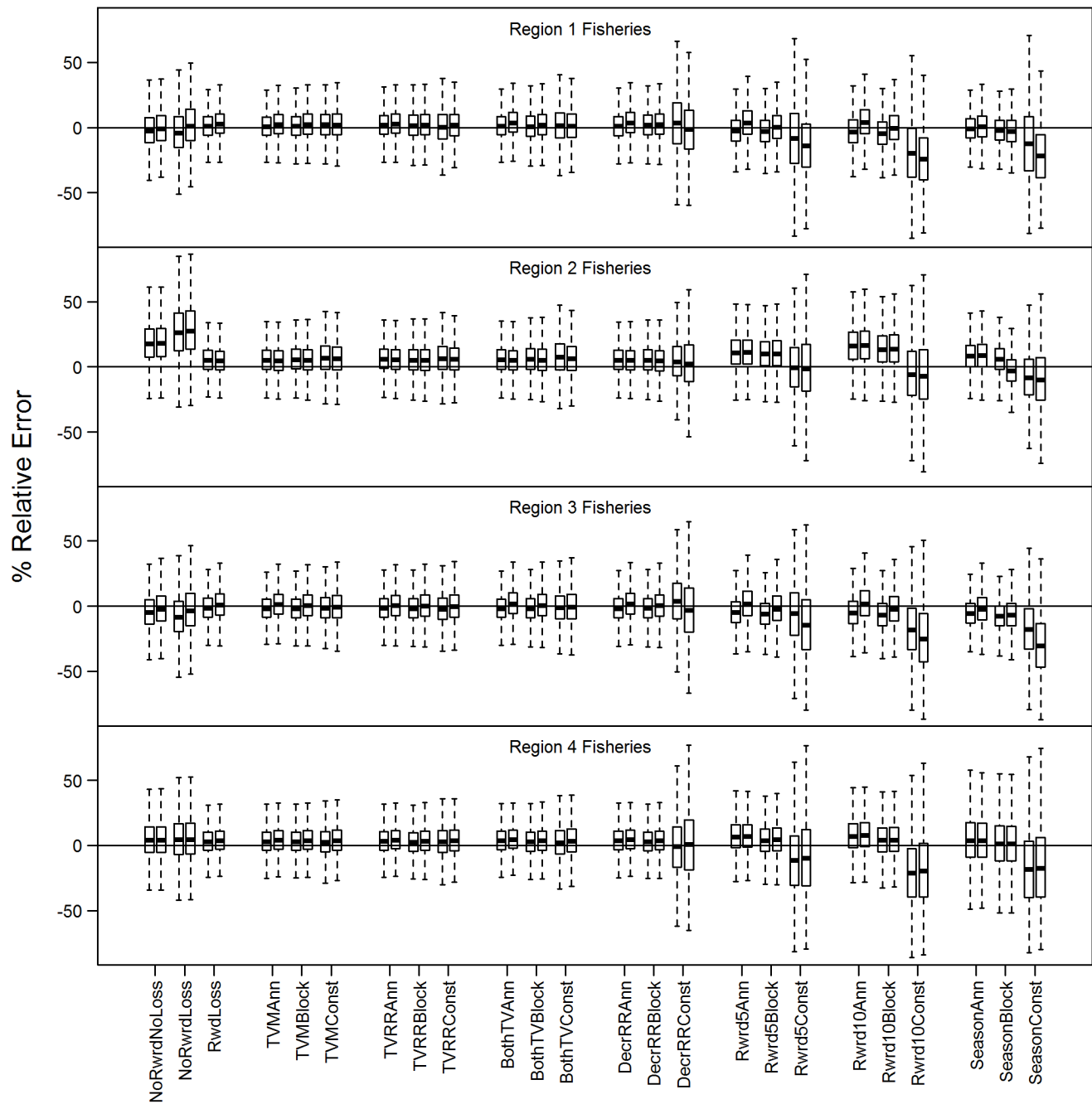


Figure 3: Same as Figure 1 except for fishery catchabilities in each region, where the first boxplot in each pair is the commercial fishery with low reporting rates and the second is the recreational fishery with higher reporting rate.

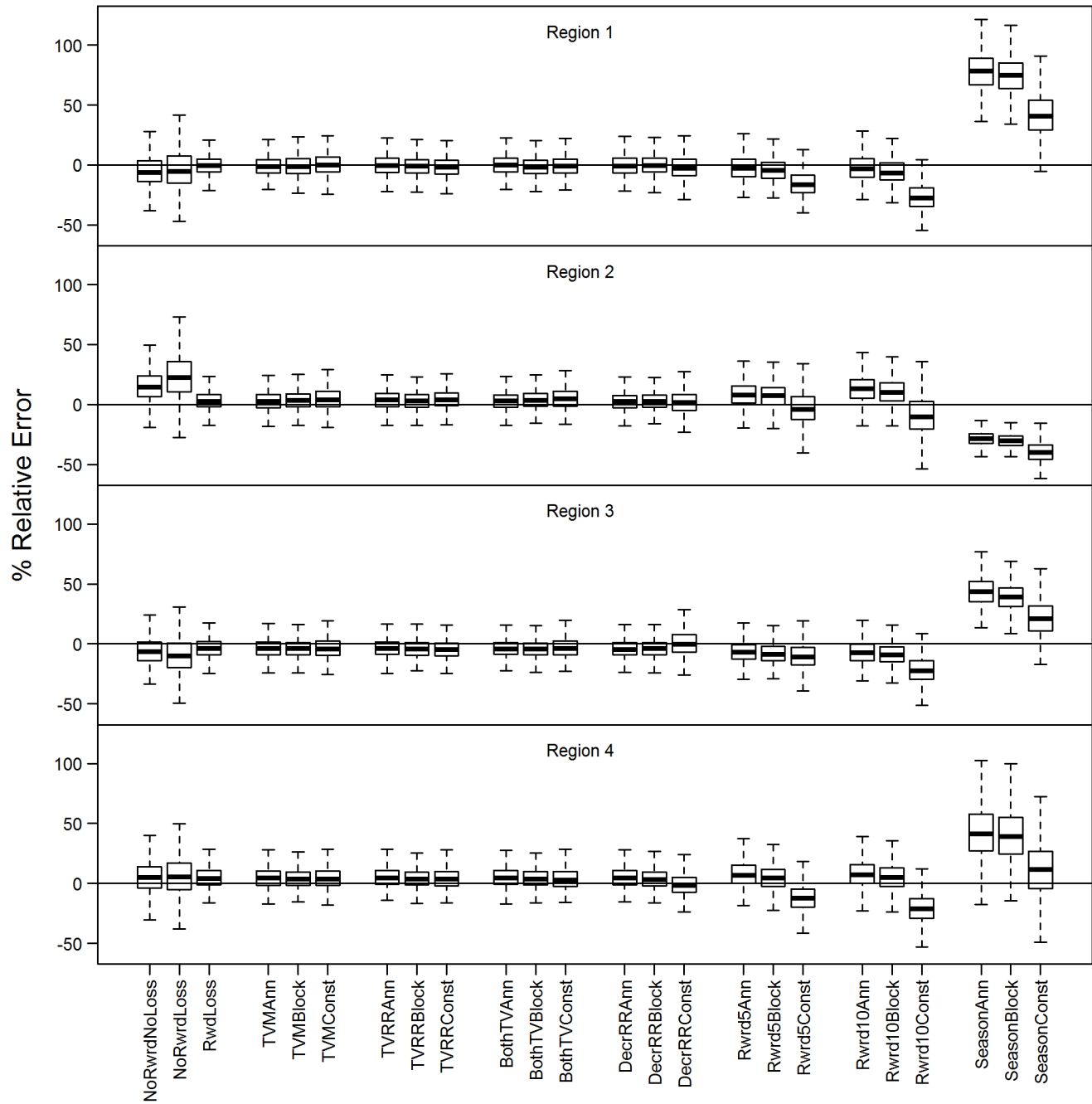


Figure 4: Same as Figure 1 except for catchability coefficients for the surveys.

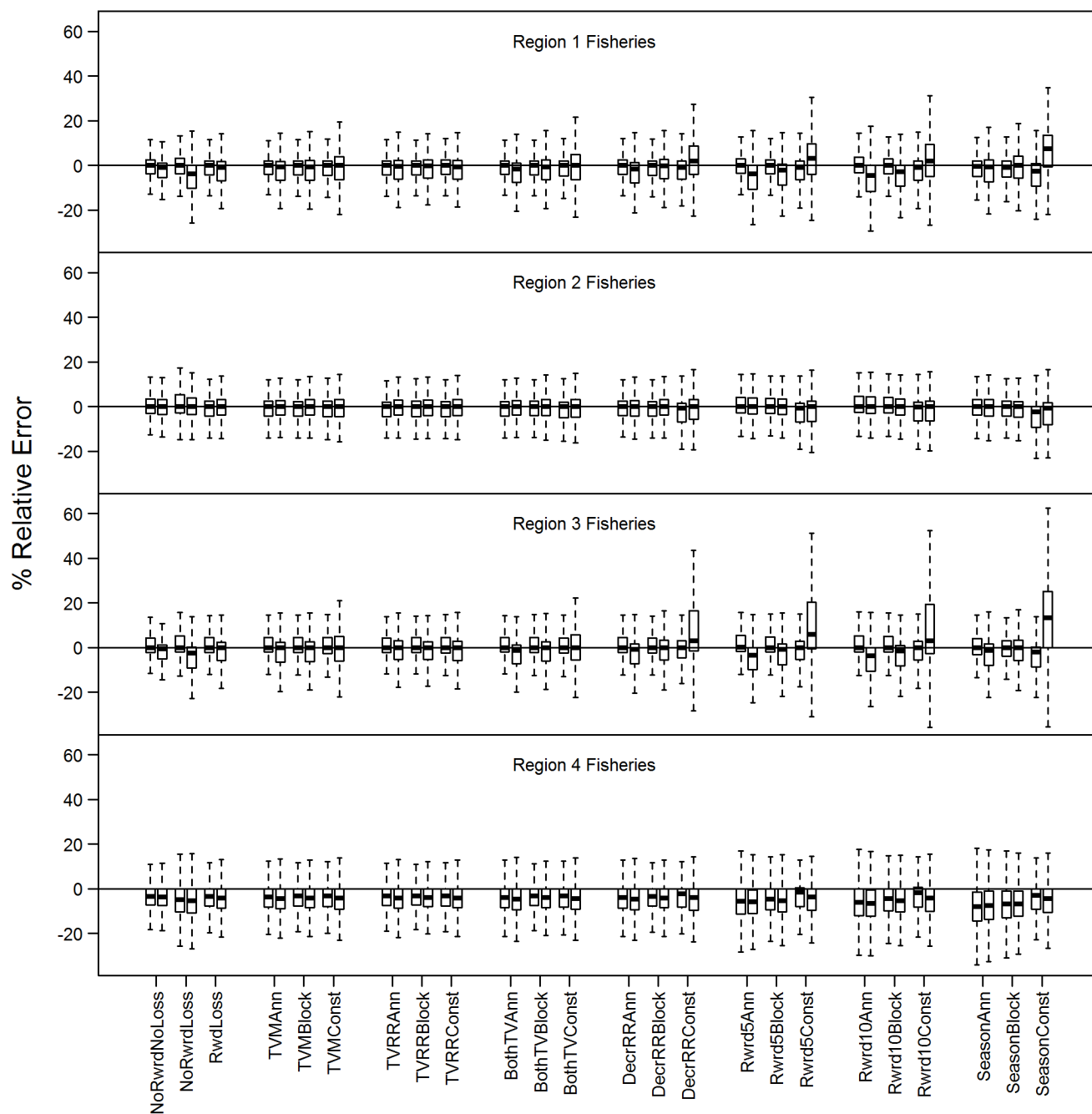


Figure 5: Same as Figure 1 except for fishery selectivity, where the first boxplot in each pair is the commercial fishery and the second is the recreational fishery.

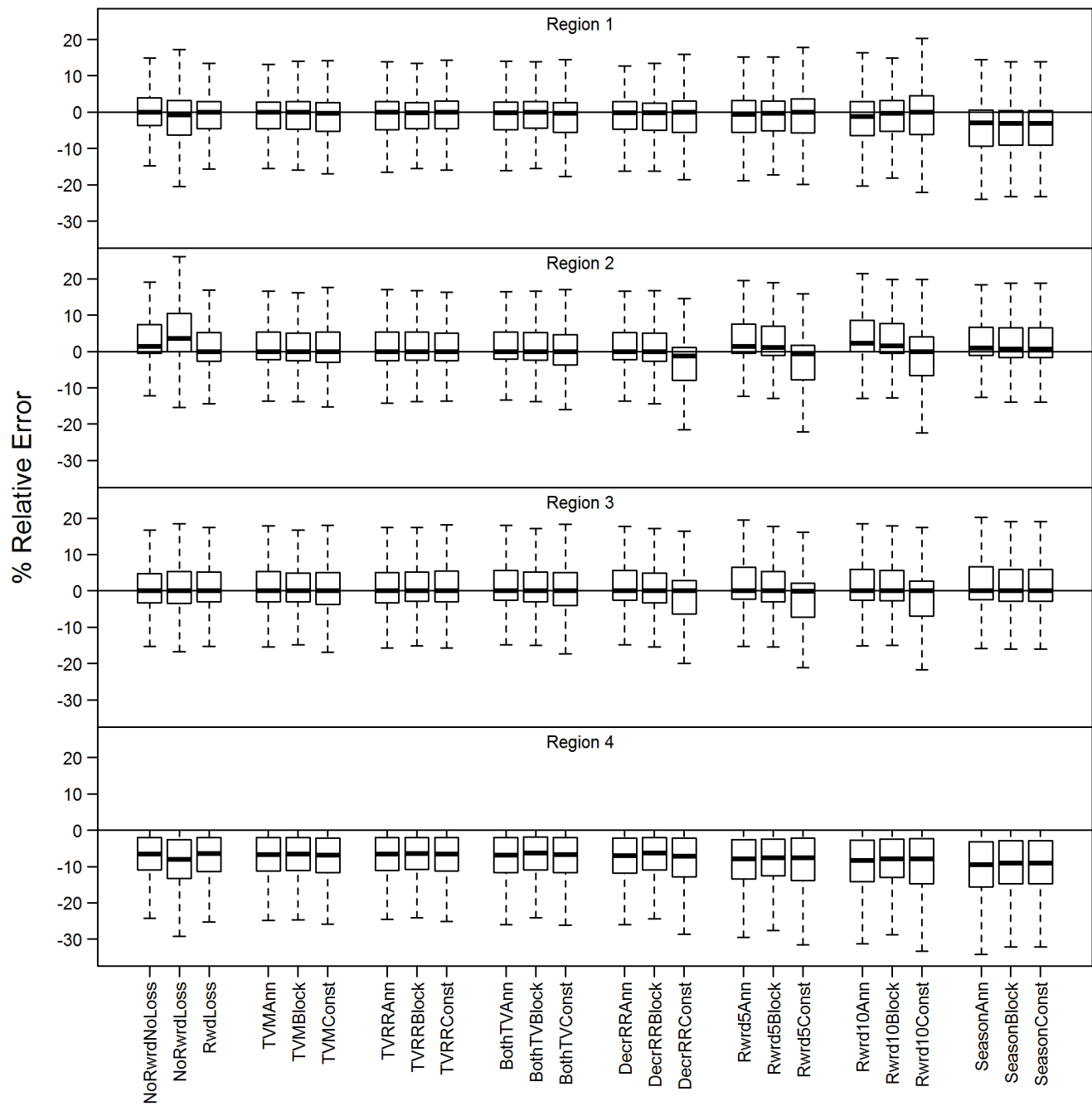


Figure 6: Same as Figure 1 except for survey selectivity.

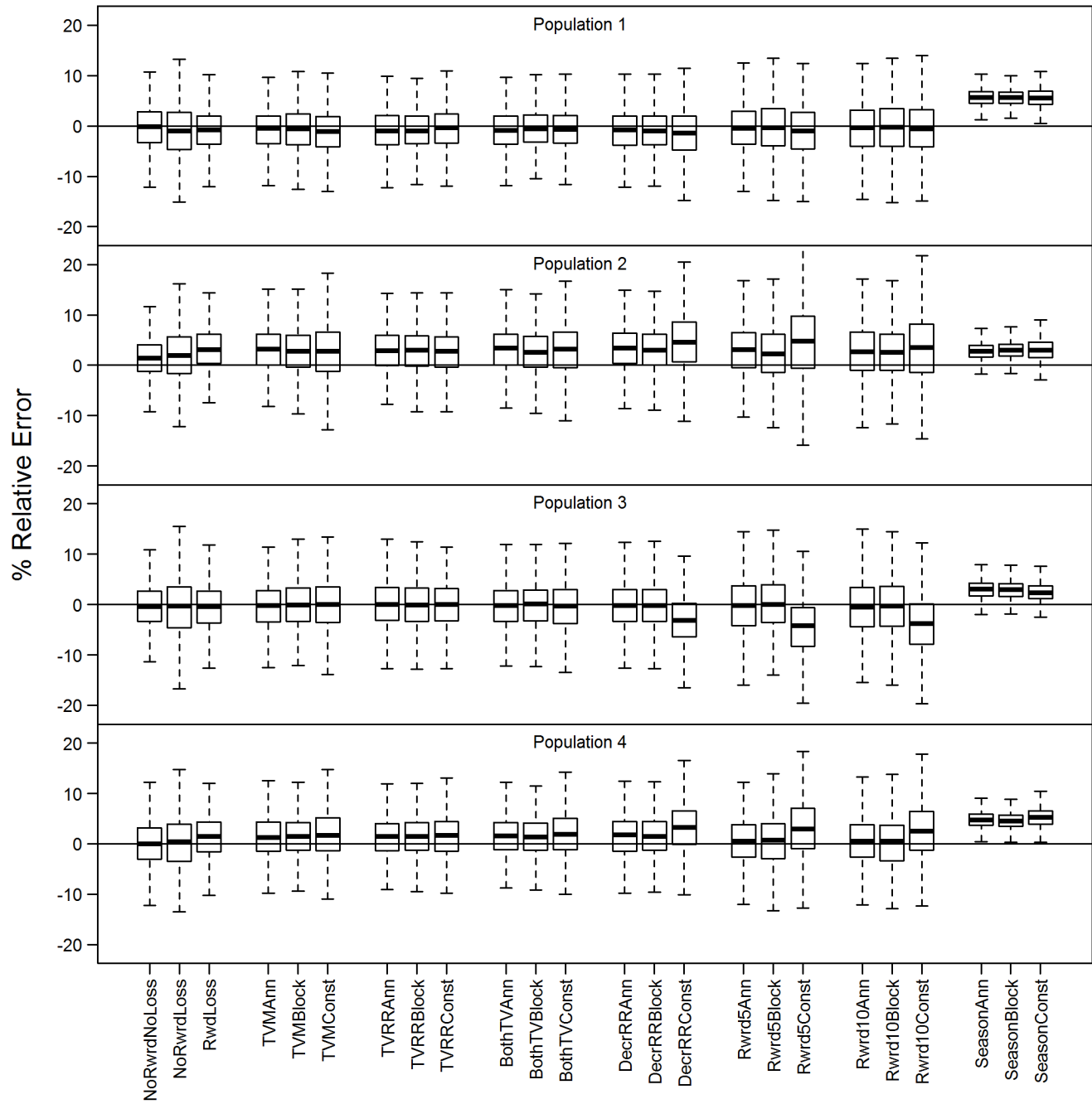


Figure 7: Same as Figure 1 except for percent of population that remains in the natal region (Stay Rate) for each population.

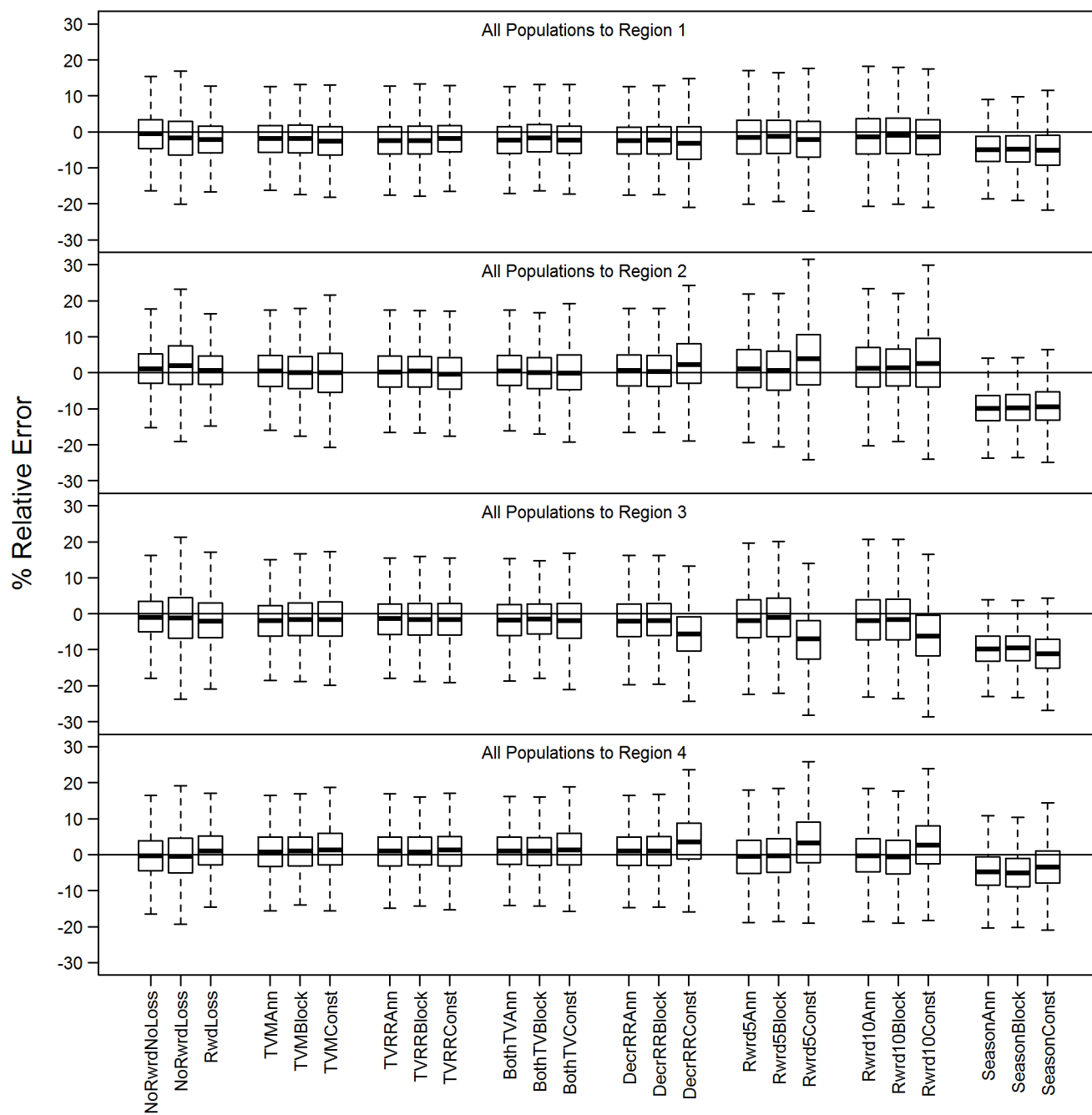


Figure 8: Same as Figure 1 except for the percentage in population that remains in the natal region (Stay Rate) for each population.

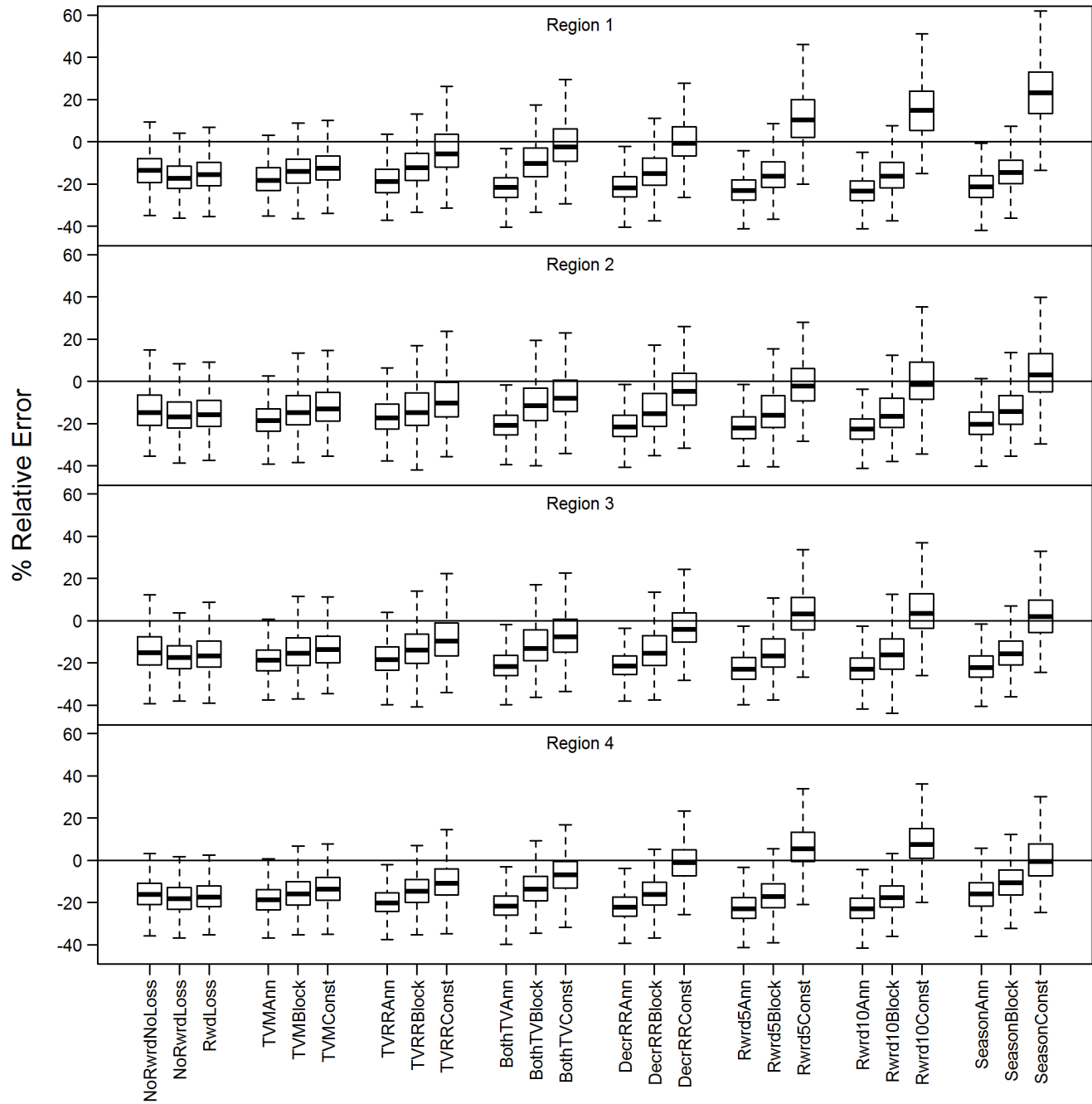


Figure 9: Same as Figure 1 except for the log-harvest variance.

Code

Copyright (c) < 2017 > Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Operating Model Code

```
library(truncnorm)
library(methods)
args=commandArgs(trailingOnly=TRUE)
args=as.numeric(args)
if (length(args)<9){stop ("You must put in atleast 9 arguments for the program to run successfully"
)}
if (length(args)==9){args[10]=4;args[11]=0;args[12]=0}
if (length(args)==10){args[11]=0;args[12]=0}
if (length(args)==11){args[12]=0}

#Put in safety checks so that the correct simulation types are called
if (!(args[2]%in%1:6)) stop("RR Generation Type (args[2]) must equal 1 to 6")
if (!(args[3]==1 | args[3]==2 | args[3]==3)) stop("RR Estimation Type (args[3]) must equal 1,2 or 3
")
if (args[3]%in%c(3) & (args[4]<=0 | args[5]<=0)) stop("Cannot have Annually varying reporting rate
estimation if the phase of estimation is negative")
if (args[3]%in%c(2) & (args[4]>=0 | args[5]<=0)) stop("If estimating Block reporting rate you must
set the phase of estimation for RR parameters negative and RRDevs to positive")
if (!(args[6]==1 | args[6]==2)) stop("M Generation Type (args[6]) must equal 1 or 2")
if (!(args[7]==1 | args[7]==2 | args[7]==3)) stop("M Estimation Type (args[7]) must equal 1,2 or 3"
)
if (args[7]%in%c(2,3) & (args[8]>=0 | args[9]<=0)) stop("LogM Phase must be negative and LogMDevs
must be positive for time varying estimations")
if (!(args[10]==4 | args[10]==8)) stop("Number of fisheries (args[6]) must equal 4 or 8")
if (!(args[11]%in%0:1)) stop("Tag Loss Type (arg[11]) must equal 0 or 1")
if (!(args[12]%in%0:3)) stop("High Reward release Type (arg[12]) must equal 0, 1, 2, or 3")
#Read in commandline arguments there should be ### of them the order of them is Movement Type,
Reporting Rate Generation Type, Reporting Rate Estimation type, Reporting Rate Estimation Phase
, Natural Mortality Generation Type, Natural Mortality Estimation Type, Natural Mortality
Estimation Phase, Number of fisheries , Tag Loss Type, High Reward Release Type,

#This code is to create data to be used in the assessment model
#Data to be created includes: annual catch data, effort data/CPUE, index of abundance?, tag return
data, annual age composition data, high reward tag return data

##### Scenario options

##Movement Scenarios 1: Base Case 70% stay 10% leave 2: 97% stay 1% leave 3: 85% stay 5% leave 4:
40% stay 20% leave 5: movement matrix 1 6: movement matrix 2 7:No movement 8: Lake Erie
9:Other?
MvmntType=args[1]
##Reporting Rate Scenarios 1: constant Reporting rate 50% spatially constant 2: randomly varying
AR(1) process with mean 50% different in each region
RRType=args[2]
##Reporting Rate Estimation Type 1: constant value through time series 2: 5 Year block estimated 3:
Random walk estimation with yearly estimates
RREst=args[3]
##Reporting Rate is it estimated? If value is positive it is the phase that the parameter is
estimated in the model, if negative and constant then it is a known value
PhaseRR=args[4]
##Time Varying Reporting Rate is it estimated?
RRVaryPhase=args[5]
```

```

##Natural Mortality generation options 1: Constant over the time series 2: Autoregressive 1 process
  generates yearly values
MType=args[6]
##Natural Mortality estimation options 1: Constant 2: 5Year Block estimation 3: Random walk
  estimation with yearly estimates
MEst=args[7]
#Natural mortality phase of estimation
PhaseM=args[8]
#Time Varying Natural mortality phase of estimation
MVaryPhase=args[9]
#Number of fisheries. If there are same number of fisheries as region then there is one in each,
  if there is double there is two in each. Commercial in each region and then recreational in
  each region if two fisheries.
fisheries=args[10]
#Tag Loss Scenarios 0: No tag loss 1: different tag retention in each region of release, which
  decreases each year after release
TagLossType=args[11]
#High Reward Scenarios 0: No High value tags released 1: High value tags released every year 2:
  High value tags released every 5 years 3: High value tags released every 10 years
HighRewardType=args[12]

#How often the Natural Mortality Parameter varies Should keep this at 5 since tpl is set for this
YrsMVary=5
##Time Varying Movement Phase of estimation
MvmntVaryPhase=-10
## Time Varying movement scenarios 0:Not time varying 1: Randomly varying Movement 2: linealy
  increaseing movement out
MvmntTVType=0

##### PARAMETERS

#Set up parameters for total number
years=40
regions=4
stocks=4
ages=6
if (MType==1){
  M=rep(0.32,years) ## Constant M
} else if (MType==2){ ## AR(1) M
  M=numeric(years)
  #Set the autocorrealation for M based on simulating different values, these looked the most
  reasonable
  Mphi=0.8
  #Set the standard deviation that you want the stationary SD to be
  Msd=0.05
  #set the standard deviation of the random variable so that the stationary variance is equal to
  0.05^2
  Msigma=sqrt(Msd^2*(1-Mphi^2))
  M[1]=rtruncnorm(1,mean=0.32,sd=Msigma,a=0,b=Inf)
  #Calculate constant so that the mean will be the Natural mortality that we want 0.32
  c=0.32*(1-Mphi)
  #Calculate an autoregressive trend for the natural mortality
  for (y in 2:years) M[y]=c+Mphi*M[(y-1)]+rtruncnorm(1, mean=0,sd=Msigma,a=-(Mphi*M[(y-1)]+c),b=
  Inf)
}

#Vector for proportion of tags lost for each stock could be fancier if need be
#For now assume there is no tag loss
TagRetention=matrix(1,nrow=years,ncol=regions)
if (TagLossType==1){
  rhoA=c(0.99,0.95,0.95,0.99)
  phiA=c(0.08,0.28,0.28,0.07)
  for (y in 1:years){
    for (s in 1:stocks){
      TagRetention[y,s]=rhoA[s]*exp(-phiA[s]*y)
    }
  }
}

##Generate the Reporting rate based on the RRType. When RRType==1 then constant at 50% cross all
  regions and time. When RRType==2 then generate random process for each region using an AR(1)

```



```

    process with mean 0.5 and stationary variance of 0.05
if (RRType==1){ReportingRate=matrix(0.5,nrow=years,ncol=fisheries)}
if (RRType==2){
  ReportingRate=matrix(0,nrow=years,ncol=fisheries)
  #Set the mean RR for all regions
  if (fisheries==4){meanRR=rep(0.5,fisheries)}
  if (fisheries==8){meanRR=c(rep(0.15,4),rep(0.43,4))}
  #set what the standard deviation of the stationary variance is
  sdRR=0.05
  #Set the autocorrelation level based on test plots
  RRphi=0.7
  #Calculate the SD for the white noise random error
  RRsigma=sqrt(sdRR^2*(1-RRphi^2))
  #Calculate constant so that mean is close to 0.5
  c=meanRR*(1-RRphi)
  #Generate Starting value based on a truncated normal distribution
  ReportingRate[1,]=rtruncnorm(fisheries,meanRR,RRsigma,a=0,b=1)
  #Generate the time series using an AR(1) process
  for (f in 1:fisheries){
    for (y in 2:years){
      ReportingRate[y,f]=c[f]+RRphi*ReportingRate[(y-1),f]+rtruncnorm(1,mean=0,sd=RRsigma,a=-(
        RRphi*ReportingRate[(y-1),f]+c[f]),b=1-(c[f]+RRphi*ReportingRate[(y-1),f]))
    }
  }
}
if (RRType==3){ReportingRate=matrix(c(0.3,0.7,0.1,0.5),nrow=years,ncol=fisheries,byrow=TRUE)}
if (RRType==4){ReportingRate=matrix(c(0.5,0.1,0.7,0.3),nrow=years,ncol=fisheries,byrow=TRUE)}
if (RRType==5 & fisheries==8){ReportingRate=matrix(c(0.25,0.15,0.2,0.1,0.35,0.4,0.3,0.45),nrow=
  years,ncol=fisheries,byrow=TRUE)}else if (RRType==5 & fisheries==4){stop("Must be 8 fisheries")}
if (RRType==6 & fisheries==8){ReportingRate=matrix(c(seq(from=.25,to=.05,length.out=40),seq(from=.2,to=.1,length.out=40),seq(from=.2,to=.05,length.out=40),
  seq(from=.5,to=.25,length.out=40),seq(from=.55,to=.2,length.out=40),seq(from=.5,to=.15,length.out=40),seq(from=.60,to=.20,length.out=40)),nrow=years,ncol=fisheries)}else if (RRType==6 &
  fisheries==4){stop("Must be 8 fisheries")}

## Set the true value of reporting rate
rr=ReportingRate[1,]

#Set the fishing mortality rate for each area and each fishery
effort = array(0,dim=c(years,fisheries))
##Create random fishing mortality trends using an AR(1) process that is estimated from the Western
  Basin fully selected fishing mortality. The means for the different regions are calculated
  based on the estimated values from the fully selected age and the central basin is assumed to
  have the same mean as the eastern Basin
meanFs=numeric(length=fisheries)

if (fisheries==4){
  #Calculate the means of the fishing mortalities for the 4 regions
  #Lake Huron Total Fishing mortality
  meanFs[1]=mean(c
    (0.174425,0.589382,0.0872441,0.0700667,0.0731005,0.566263,0.943108,0.107766,0.103648,0.182409,1.13299,0.16
  ))
  #Western Lake Erie Total Fishing mortality
  LakeErieF=c(0.375544, 0.496701, 0.378103, 0.45361, 0.360045, 0.239472, 0.19751, 0.156237,
    0.199569, 0.193022, 0.218706, 0.172667, 0.152607, 0.133225, 0.16557, 0.224292, 0.198858,
    0.239118, 0.295636, 0.248441, 0.314397, 0.232081, 0.207921, 0.124245, 0.0870103, 0.0938314,
    0.0748686, 0.0846649, 0.090858, 0.0987179, 0.0811872, 0.0767647, 0.0718532, 0.0668829,
    0.107389, 0.116998, 0.15464)
  meanFs[2]=mean(LakeErieF)
  #Eastern Lake Erie Total Fishing mortality
  meanFs[3]=mean(c
    (0.176891,0.360011,0.181095,0.380324,0.186269,0.205295,0.252238,0.344163,0.239906,0.158851,0.194252,0.0721
  ))
  #Use the same mean for Central Lake Erie as Western Lake Erie Total Fishing mortality
  meanFs[4]=meanFs[2]
  ##Calculate the AR(1) process from the Western basin fishing mortalities
  z=ar(LakeErieF,FALSE,order.max=1)
  #Set the autocorrelation for the processes
  Fphi=z$ar
  #Calculate the constant that needs to be added so that the mean is that of the regions
  cons=meanFs*(1-Fphi)

```

18

```

    for (y in 2:years){
      #Randomly generate a Fishing mortality schedule using an AR(1) process but used
      truncated normal distributions so that negative values are not generated
      #Total Fishing mortality Lake Huron
      effort[y, f]=cons[f]+Fphi*effort[(y-1), f]+rtruncnorm(1, mean=0, sd=Fsigma, a=-(Fphi*effort[(y-1),
      f]+cons[f]), b=Inf)
    }
  }
}

if (any(effort <=0)) stop("Apical Fishing mortality generated a value that is less than or equal to
zero")

#Create an indicator variable for if a fishery is active in region
#For now will assume all years active
FisheryActive=array(0, dim=c(fisheries, regions))
#Create an loop instead of putting in data individuall will need to when doing seperate fisheries
for( i in 1:regions){FisheryActive[i, i]=1}
if (fisheries>regions){for( i in 1:regions){FisheryActive[i+regions, i]=1}}
#Set the selectivity at age for each fishery for each area

#selectivity for L Huron based on trapnet selectivity, western L Erie based on Ontario commercial,
central L Erie based on Ohio2west (recreational fishery), eastern L Erie Ontario gill net if 4
fisheries active
if (fisheries==4){selectivity=array(c(0.35,0.98,1,0.7,0.5,0.5, 0.4,1,0.9,0.8,0.8,0.7,
0.1,0.6,0.65,0.7,0.8,1, 0.01,0.13,0.35,1,1,1), dim=c(ages, fisheries))}
if (fisheries==8){
selectivity=array(c(0.35,0.98,1,0.7,0.5,0.5, #LakeHuron trapnet
0.4,1,0.9,0.8,0.8,0.7, #Ontario commercial
0.4,1,0.9,0.8,0.8,0.7, #central Lake Erie commercial same??
0.01,0.13,0.35,1,1,1, #eastern Lake Erie gill net

0.04,0.5,0.75,.85,.92,1, #Lake Huron recreational
0.5,0.9,1,0.7,0.65,0.6, #Michigan 1 recreational
0.1,0.6,0.65,0.7,0.8,1, #Ohio2 west recreational
0.01,0.15,1,1,1,1 #NY recreational
), dim=c(ages, fisheries))
}
#selectivity for L Huron commercial, western L Erie commercial, central L Erie commercial, eastern
L Erie commercial gillnet, L Huron recreational based on recreational scaled to 7 ages max,
western L Erie recreational based on Ohio west 2, central L Erie recreational, eastern L Erie
NYPA recreational anglers

#selectivity for Survey in L Huron, western L Erie based on Ontario CPUE survey, central L Erie is
based on the Ohio cpue western basin survey and eastern L Erie assumes all ages are fully
selected in that order
SurveySel=array(c(0.6,0.7,1,0.9,0.9,0.9, 1,0.8,0.6,0.55,0.55,0.3, 1,0.5,0.4,0.3,0.3,0.3,
1,1,1,1,1,1), dim=c(ages, regions))

#Create vector of the ages that are fully recruited to the respective gears. Make sure that the age
is one less than the actual age because age 2 is a=1
FisheryFullySelected=numeric(fisheries)
SurveyFullySelected=numeric(stocks)
#Calculate what the index for the maximum selectivity value in each row of the selectivity maxtrix
This will give a warning that the number of items to replace is not a multiple of replacement
length if there is more than one age that is fully selected. This is okay because you just
want the first age that is fully selected.
for(f in 1:fisheries) FisheryFullySelected[f]=which.max(selectivity[,f])
for(s in 1:stocks) SurveyFullySelected[s]=which.max(SurveySel[,s])

#Set the initial population abundance at age for each area
if (stocks>=1){NOR1 = c(9000000,7000000,5000000,3000000,1000000,1500000)}
if (stocks>=2){NOR2 = c(1000000,800000,600000,400000,200000,90000)}
if (stocks>=3){NOR3 = c(500000,300000,100000,80000,50000,50000)}
if (stocks>=4){NOR4 = c(500000,300000,100000,80000,60000,90000)}
if (stocks>=5){NOR5 = c(50000,25000,10000,5000,1000,2500)}

#Set the movement rate between each area
#Rows indicate the region fish are coming FROM
#Columns indicate the region fish are moving TO

```

```

#Movement [FROM,TO]
if (MvmntType==1) Movement=matrix(data=c(.7,.1,.1,.1, .1,.7,.1,.1, .1,.1,.7,.1, .1,.1,.1,.7),
  nrow=regions, byrow=TRUE)
if (MvmntType==2) Movement=matrix(data=c(.97,.01,.01,.01, .01,.97,.01,.01, .01,.01,.97,.01,
  .01,.01,.01,.97), nrow=regions, byrow=TRUE)
if (MvmntType==3) Movement=matrix(data=c(.85,.05,.05,.05, .05,.85,.05,.05, .05,.05,.85,.05,
  .05,.05,.05,.85), nrow=regions, byrow=TRUE)
if (MvmntType==4) Movement=matrix(data=c(.4,.2,.2,.2, .2,.4,.2,.2, .2,.2,.4,.2, .2,.2,.2,.4),
  nrow=regions, byrow=TRUE)
if (MvmntType==5) Movement=matrix(data=c(.95,.05,0,0,.15,.55,.25,.05,.02,.07,.8,.11,0,.02,.12,.86),
  nrow=regions, byrow=TRUE)
if (MvmntType==6) Movement=matrix(data=c(.97,.03,0,0,.08,.78,.12,.02,.01,.03,.9,.06,0,.01,.06,.93),
  nrow=regions, byrow=TRUE)
if (MvmntType==7) for(i in 1:regions) Movement=matrix(c(1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1),nrow=
  regions, byrow=TRUE)
if (MvmntType==8) Movement=matrix(data=c(.75,.12,.08,.05, .07,.8,.08,.05, .03,.06,.87,.04,
  .02,.04,.06,.88), nrow=regions, byrow=TRUE)

#perform test to make sure that the rows sum to 1 so that no fish are created
checkSums=rowSums(Movement)
eps=1
while ((eps+1)>1){eps=0.5*eps}
for (r in 1:regions){if (!(checkSums[r]<(1+2*eps) && checkSums[r]>(1-2*eps))){stop("Movement does
  not sum to 1")}}}

#Set the parameters for the recruitment curve
alpha = c(2.41807, 1.48449, 1,0.34915)
beta = c(1.29135e-6, 3.0618e-8, 1e-6,2.80287e-8 )

#Set the maturity schedule to use in the Ricker equation for recruitment
#Below are values from the western basin assessment
maturity=c(0.308,0.824,0.914,0.935,.978,1)
weight=c(0.8347,1.1659,1.4875,1.7687,1.944,2.3323)

#set the CV for the Initial Abundance deviations
RandomCV=0.3

#set the CV for the observation error in the observed datasets
catchCV = 0.1
effortCV=0.1
processCV=0.04
surveyCV=0.2

#Create array to keep track of the temporal correlation for each stock
#This value comes from Thorson et al 2014 The estimate for perciformes the Autocorrelation from
  table 2 of 0.466 with a SD of 0.260
#Randomly simulate the autocorrelation based on the posterior distribution mean and sd
rho=rtruncnorm(4,mean=0.466,sd=0.260,a=-0.99,b=0.99)

#Calculate what the variability needs to be to get stationary variance with the autocorrelation
  term
logrecruitCV=list(mu=NA,sd=NA)
#Randomly generate the recruitment CV based on the estimated SD from Thorson et al 2014. This does
  not need to be bias corrected or transformed from a CV because it is estimated on the log scale
  as a standard deviation
logrecruitCV$sd=rtruncnorm(4,mean=0.777,sd=0.313,a=0,b=Inf)
logrecruitCV$mu=-(.5*logrecruitCV$sd^2*(1-rho)/sqrt(1-rho^2))

#Function to calculate what the mean and the standard deviation should be for the lognormal
  distribution given the mean and CV on the normal scale
lognormmud <- function(mean,CV) {
  sigsq=log(CV^2+1)
  mu=log(mean)-(.5*sigsq)
  result=list(mu=mu,sd=sqrt(sigsq))
  return(result)
}

#Calculate the mean and sd for the random variables to be input into rlnorm functions

```

```

logcatchCV=lognormmud(1,catchCV)
logeffortCV=lognormmud(1,effortCV)
logprocessCV=lognormmud(1,processCV)
logsurveyCV=lognormmud(1,surveyCV)
logRandomCV=lognormmud(1,RandomCV)

Test=array(0,dim=c(151,ages,stocks))
if (stocks>=1) Test[1,,1]=N0R1;
if (stocks>=2) Test[1,,2]=N0R2;
if (stocks>=3) Test[1,,3]=N0R3;
if (stocks>=4) Test[1,,4]=N0R4;
if (stocks>=5) Test[1,,5]=N0R5

for(y in 1:150){
  for(s in 1:stocks){
    Test[(y+1),1,s]= alpha[s]*((maturity*weight)%*%Test[y,,s])*exp(-beta[s]*((maturity*weight)%
    *%Test[y,,s]))
    for(a in 1:(ages-1)){
      Test[(y+1),(a+1),s] = Test[y,a,s]*exp(-M[1])
    }
    Test[(y+1),ages,s]=Test[y,(ages-1),s]*exp(-M[1])+Test[y,ages,s]*exp(-M[1])
  }
}
StartPop=Test[151,,]

#Create an array for the abundance through time in each area
N=array(0, dim=c((years+2),ages,stocks))
#Set the initial population sizes as the equilibrium for the recruitment functions without movement
  but add in random variation to the ages
N[1,,]=StartPop*rlnorm(n=length(StartPop),meanlog=logRandomCV$mu, sdlog=logRandomCV$sd)
#Start Autocorrelation value for the second year of recruitment
Autocorrelation=array(dim=c(years+1,stocks))
Autocorrelation[1,]=rlnorm(n=stocks,meanlog=logrecruitCV$mu, sdlog=logrecruitCV$sd)
#Calculate the Recruitment for the second year with the first random value of autocorrelation. Need
  to do this here because of the two year lag on recruitment
N[2,1,]=StartPop[1,]*Autocorrelation[1,]

#set the sample size for the age composition data simulation
AgeCompSamples=array(100,dim=c(years,ages,fisheries))
SurveyESS=array(100,dim=c(years,ages,regions))

#Create array to store the fish in after they have moved
#The stock is the area from which the fish originated from and the region is the area to which is
  moves post spawning at the beginning of the year
NMvmt=array(0,dim=c(years,ages,stocks,regions))
#Create array for the total catch in each region
CatchAge=array(0,dim=c(years,ages,fisheries))
TotalCatch=array(0,dim=c(years,fisheries))

##### Abundance Calculations
#calculate the population abundance for the 5 populations based upon the above parameters

#Let the following letter ber used for loops
# a is the age of the fish 2:7 in reality but just use 1:6 for calculations
# y is the year 1:40
# r is the region 1:5 in which the fish is residing
# f is the fisheries (for now just one)
# s is the stock from which the fish originates. For now we are assuming that the number of
  regions is the same as the number of stocks

#Calculate arrays for F, Z and Surv
F=array(0,dim=c(years,ages,regions,fisheries))
FTotal=array(0,dim=c(years,ages,regions))
Z=array(0,dim=c(years,ages,regions))
# FFull=array(0,dim=c(years,fisheries))
FFull=effort

for(f in 1:fisheries){
  #Apply process error to the underlying apical F
  # FFull[,f]=effort[,f]*rlnorm(length(effort[,f]),logprocessCV$mu,logprocessCV$sd)
  for(r in 1:regions){
    #Calculatethe age and region specific fishing mortality

```

```

F[, , r, f] = (FFull[, , f] %*% t(selectivity[, , f])) * FisheryActive[f, r]
#Calculate the total fishing mortality within each region by summing over active fisheries
FTotal[, , r] = FTotal[, , r] + F[, , r, f]
}
}

#Add natural mortality to fishing mortality
for(y in 1:years) Z[y, ,] = FTotal[y, ,] + M[y]
#Convert Z to survival for easier use
Survival = exp(-Z)

SurveyAge = array(0, dim=c(years, ages, regions))
#Survey Catchability coefficient for L Huron based on Saginaw Bay survey, western L Erie based on
  Ohio CPUE, central L Erie CPUE taken from western basin ontario gill net Q, eastern L Erie NY
  net CPUE survey
qSurvey = c(1.5e-5, 5e-6, 2e-7, 8e-7)

#Begin loop over all of the years
for(y in 1:years){
  #Begin loop for each area
  for(s in 1:stocks){
    # simulate the recruitment for age 2 for each stock with a temporal autocorrelation so
    there is a 2 year time lag on recruitment but age 2 is the first age in model
    #This is y+1 because first value was filled in earlier from the equilibrium stock
    Autocorrelation[y+1, s] = rho[s] * Autocorrelation[y, s] + rnorm(n=1, mean=logrecruitCV$mu[s], sd=
logrecruitCV$sd[s]) * sqrt(1-rho[s]^2)
    N[(y+2), 1, s] = alpha[s] * ((maturity*weight)%*%N[y, , s]) * exp(-beta[s] * ((maturity*weight)%*%N[y
, , s]) + Autocorrelation[(y+1), s])
    if (sum(N[y, , s]) < 40000) {
      message("This run through had a population that is less than 40000")
      source("DataSimulator.r")
      #Stop after rerunning to make sure that it doesn't rerun at the end
      stop()
    }
    if (N[y+2, 1, s] < 100) {
      message("This run through had recruitment that is less than 100")
      source("DataSimulator.r")
      #Stop after rerunning to make sure that it doesn't rerun at the end
      stop()
    }
    if (N[y+2, 1, s] > 1e10) {
      message("This run through had a recruitment greater than 1e10")
      source("DataSimulator.r")
      #Stop after rerunning to make sure that it doesn't rerun at the end
      stop()
    }
  } #End stock loop

  #Begin loop over ages
  for(a in 1:ages){
    for(r in 1:regions){
      for(s in 1:stocks){
        #Calculate the number of fish that move to each area from spawning area and apply
        mortality
        NMvmnt[y, a, s, r] = N[y, a, s] * Movement[s, r] * Survival[y, a, r]
        SurveyAge[y, a, r] = SurveyAge[y, a, r] + N[y, a, s] * Movement[s, r] * exp(-Z[y, a, r] * 10/12) *
        SurveySel[a, r] * qSurvey[r]
        #Calculate the catch for each area with ages separate
        #Need to sum over the different spawning stocks
        #C=F/Z*(N*(1-surv))
        for(f in 1:fisheries){
          CatchAge[y, a, f] = CatchAge[y, a, f] + ((F[y, a, r, f] / Z[y, a, r]) * (N[y, a, s] * Movement[s, r])
          * (1 - Survival[y, a, r]))
        }
        #Calculate those that survive to the next year to spawn for each stock
        if(a < ages){
          N[(y+1), (a+1), s] = N[(y+1), (a+1), s] + NMvmnt[y, a, s, r]
        } else{
          N[(y+1), ages, s] = N[(y+1), ages, s] + NMvmnt[y, ages, s, r]
        }
      }
    }
  }
}

```

```

    }
  }
  #End stock loop
}
#End region loop
for (f in 1:fisheries){
  #Sum the catch over ages in each area
  #Need to do this outside of stock loop and region loop or results in over counting the
  catch
  TotalCatch[y, f]=TotalCatch[y, f]+CatchAge[y, a, f]
}
}
#End age loop
}
#End year loop
##### Data Simulation

#Add lognormal observation error to catch in each area
ObservedCatch=TotalCatch*rlnorm(n=length(TotalCatch), meanlog=logcatchCV$mu, sdlog=logcatchCV$sd)
#Fishery Catchability coefficient for L Huron based on gill net catchability, western L Erie based
  on commercial catchability, central L Erie fishery based on q for Ohio recreation fishery and
  eastern L Erie based on Mu4 commercial fishery in that order
if (fisheries==4){q=matrix(c(2e-6,8e-6,3e-5,6e-5),nrow=years, ncol=fisheries, byrow=TRUE)}
if (fisheries==8){
q=matrix(c(
2e-6, #L Huron gillnet
5e-6, #Ontario commercial from western basin assessment
6e-7, #Ontario gillnet from eastern basin assessment
8e-7, #NY net

2e-8, #Lake Huron recreational
1e-5, #Michigan recreational
3e-5, #Ohio recreational
3e-7 #NYPA recreational
),nrow=years, ncol=fisheries, byrow=TRUE)
}

#Create Arrays to store the observed CPUE survey and age composition proportion for each region
ObservedSurvey=array(NA,dim=c(years,regions))
ObservedSurveyAgeComp=array(NA,dim=c(years,ages,regions))
#Add lognormal observation error to the calculated survey index and apply catchability coefficient
for (y in 1:years){
  for (r in 1:regions){
    ObservedSurvey[y, r]=sum(SurveyAge[y, , r])*rlnorm(1, logsurveyCV$mu, logsurveyCV$sd)
    ObservedSurveyAgeComp[y, , r]=rmultinom(1, SurveyESS[y, 1, r], SurveyAge[y, , r])/SurveyESS[y, , r]
  }
}

#Add lognormal observation error to the fishing mortality with a catchability coefficient
ObservedEffort=FFull/q*rlnorm(n=length(FFull), meanlog = logeffortCV$mu, sdlog = logeffortCV$sd)
#Simulate tag recoveries from multivariate distribution
ObservedAgeComp=array(0,dim=c(years,ages,fisheries))
#Simulate age composition from multivariate distribution of catches and turn into a proportion
for (y in 1:years){
  for (f in 1:fisheries){
    ObservedAgeComp[y, , f]=rmultinom(n=1, size=AgeCompSamples[y, 1, f], prob=CatchAge[y, , f])/
    AgeCompSamples[y, , f]
  }
}

#####Tagging Data simulator

#Number released each year in each region
TagsReleased=matrix(2000,nrow=years, ncol=stocks, byrow=TRUE)

#Assume that there is the same proportion of ages from each release in each region
ProportionRelease=c(.05,.1,.2,.2,.2,.25)
ReleaseAge=array(0,dim=c(years,ages,stocks))
TagsAlive=array(0,dim=c(years,(years+1),ages,stocks))

#This keeps track of the tagged fish that are alive at the beginning of each year. Thus it starts
  out as the number of released by age in region for each year of release.

```



```

#year of release , year of recapture (or current year concerned about), age ,stock released from
for(y in 1:years){
  for(s in 1:stocks){
    ReleaseAge[y,,s]=round( TagsReleased[y,s]*ProportionRelease)
    ReleaseAge[y,ages,s]=TagsReleased[y,s]-sum(ReleaseAge[y,-ages,s])
    TagsAlive[y,y,,s]=ReleaseAge[y,,s]
  }
}

#Create matrix to calculate where fish are after movement each year
TagMvmnt=array(0,dim=c(years,years,ages,stocks,regions))

#Create vector to store fate of tagged fish in a region from each release
TagFate=array(0,dim=c(years,years,ages,stocks,regions,(fisheries+2)))
#Caught by fisheries , natural mortality , survival
#Create array to store the recaptured tags information
TagsRecaptured=array(0,dim=c(years,years,ages,stocks,regions,fisheries))
#release event year,recapture year, age, release stock, recapture region

#Create vector to temporarily store the probability of capture by fisheries
CaptureProb=numeric(length=(fisheries+2))

#begin loop over tagging year
for(ty in 1:years){
  #begin loop over recapture year
  for(ry in ty:years){
    #loop over ages
    for(a in 1:ages){
      #loop over release stocks
      for(s in 1:stocks){

        #Check to make sure that there are still fish alive for this release at this age
        #Tag movement to new areas and apply tag loss by removing from the sample size
        of Tags Alive
        #This needs to be outside of for loop for regions
        #Tag movement using MULTINOMIAL distribution
        TagMvmnt[ty,ry,a,s]=rmultinom(n=1,size=round(TagsAlive[ty,ry,a,s]*(
        TagRetention[(ry-ty+1),s]),prob=Movement[s,]))
        #check to make sure tags aren't created or destroyed during movement
        if(round(TagsAlive[ty,ry,a,s]*(TagRetention[(ry-ty+1),s])) != sum(TagMvmnt[ty,
        ry,a,s,]))){stop("Something does not add up in the tag movement")}
```

 #loop over recapture region

```

        for(r in 1:regions){
          if(TagMvmnt[ty,ry,a,s,r]<0)stop("negative movement!!!")
          #Calculate probability of death by natural mortality and those that survive
          CaptureProb[(fisheries+1)]=M[ry]/Z[ry,a,r]*(1-Survival[ry,a,r])
          CaptureProb[(fisheries+2)]=Survival[ry,a,r]
          #Loop over fisheries
          for(f in 1:fisheries){
            #Calculate the capture probability for each fishery
            CaptureProb[f]=F[ry,a,r,f]/Z[ry,a,r]*(1-Survival[ry,a,r])
          }
          #End fisheries loop
          #Determine tag fate using MULTINOMIAL distribution
          TagFate[ty,ry,a,s,r]=rmultinom(n=1,size=TagMvmnt[ty,ry,a,s,r],prob=
          CaptureProb)

          #store the tags that are recaptured by fishery
          TagsRecaptured[ty,ry,a,s,r]=TagFate[ty,ry,a,s,r,1:fisheries]
          #test to make sure tags aren't created or destroyed during tag fate
          calculations
          if(sum(TagFate[ty,ry,a,s,r,]) !=TagMvmnt[ty,ry,a,s,r]) stop("something not
          adding up in movement 1")
        }
        #End regions loop
        #test to make sure tags aren't created or destroyed anywhere
        if(sum(TagFate[ty,ry,a,s,,]) !=sum(TagMvmnt[ty,ry,a,s,])) stop("something not
        adding up in movement 2")
        #check to make sure that tags weren't created or destroyed
        if(round(TagsAlive[ty,ry,a,s]*(TagRetention[(ry-ty+1),s])) != sum(TagFate[ty,ry
```



```

, a, s, , ])) { stop("Something does not add up in the tagging") }
    #Progress those fish that survive to the next year and age
    #remove those fish that die from the sample size of released fish i.e. only
keep survivals
    if (a < (ages - 1)) {
        TagsAlive[ty, (ry + 1), (a + 1), s] = sum(TagFate[ty, ry, a, s, , (fisheries + 2)])
    } else if (a == ages) {
        TagsAlive[ty, (ry + 1), a, s] = sum(TagFate[ty, ry, (ages - 1), s, , (fisheries + 2)] +
        TagFate[ty, ry, ages, s, , (fisheries + 2)])
    } #End if else for plus group calculations
    } #End stocks loop
    } #End age loop
    } #End capture year loop
} #End tagging year loop

#Calculate the tag returns by summing over ages
TagReturns = colSums(aperm(TagsRecaptured, perm = c(3, 1, 2, 4, 5, 6)), dim = 1)

#reformat the Tag returns to get rid of the dimension for region of recapture
#This assumed that each fishery is only active in one region
#Also apply the reporting rate for that fishery
#This will only work if the fishery is active in only one region
TagsReported = array(0, dim = c(years, years, stocks, fisheries))
NeverRecovered = matrix(data = 0, nrow = years, ncol = stocks)
for (ty in 1:years) {
    for (ry in ty:years) {
        for (s in 1:stocks) {
            for (f in 1:fisheries) {
                tempr = which(FisheryActive[f,] == 1)
                TagsReported[ty, ry, s, f] = rbinom(1, TagReturns[ty, ry, s, tempr, f], ReportingRate[ry, f])
            }
        }
    }
}

for (y in 1:years) {
    for (s in 1:stocks) {
        NeverRecovered[y, s] = TagsReleased[y, s] - sum(TagsReported[y, , s, ])
    }
}

#####High Reward Tag Simulator
if (HighRewardType > 0) {

#Number released each year in each region
#High Reward Type 1: release 200 each year 2: release 200 every 5 years 3: release 200 every 10
years

if (HighRewardType == 1) RewardTagsReleased = matrix(200, nrow = years, ncol = stocks)
if (HighRewardType == 2) RewardTagsReleased = matrix(c(200, 0, 0, 0, 0), nrow = years, ncol = stocks)
if (HighRewardType == 3) RewardTagsReleased = matrix(c(200, rep(0, 9)), nrow = years, ncol = stocks)

#Assume that there is the same proportion of ages from each release in each region
RewardReleaseAge = array(0, dim = c(years, ages, stocks))
RewardTagsAlive = array(0, dim = c(years, (years + 1), ages, stocks))

#This keeps track of the tagged fish that are alive at the beginning of each year. Thus it starts
out as the number of released by age in region for each year of release.
#year of release, year of recapture (or current year concerned about), age, stock released from
for (y in 1:years) {
    for (s in 1:stocks) {
        RewardReleaseAge[y, , s] = round(RewardTagsReleased[y, s] * ProportionRelease)
        RewardReleaseAge[y, ages, s] = RewardTagsReleased[y, s] - sum(RewardReleaseAge[y, -ages, s])
        RewardTagsAlive[y, y, , s] = RewardReleaseAge[y, , s]
    }
}

#Create matrix to calculate where fish are after movement each year
RewardTagMvmnt = array(0, dim = c(years, years, ages, stocks, regions))

```

```

#Create vector to store fate of tagged fish in a region from each release
RewardTagFate=array(0,dim=c(years,years,ages,stocks,regions,(fisheries+2)))
#Caught by fisheries, natural mortality, survival
#Create array to store the recaptured tags information
RewardTagsRecaptured=array(0,dim=c(years,years,ages,stocks,regions,fisheries))
#release event year, recapture year, age, release stock, recapture region

#Create vector to temporarily store the probability of capture by fisheries
RewardCaptureProb=numeric(length=(fisheries+2))

#begin loop over tagging year
for(ty in 1:years){
  #begin loop over recapture year
  for(ry in ty:years){
    #loop over ages
    for(a in 1:ages){
      #loop over release stocks
      for(s in 1:stocks){

        #Check to make sure that there are still fish alive for this release at this age
        #Tag movement to new areas and apply tag loss by removing from the sample size
        of Tags Alive
        #This needs to be outside of for loop for regions
        #Tag movement using MULTINOMIAL distribution
        RewardTagMvmnt[ty,ry,a,s]=rmultinom(n=1,size=round(RewardTagsAlive[ty,ry,a,s]*
        (TagRetention[(ry-ty+1),s])),prob=Movement[s,])
        #check to make sure tags aren't created or destroyed during movement
        if(round(RewardTagsAlive[ty,ry,a,s]*(TagRetention[(ry-ty+1),s])) != sum(
        RewardTagMvmnt[ty,ry,a,s,])){stop("Something does not add up in the reward tag movement")}

        #loop over recapture region
        for(r in 1:regions){
          if(RewardTagMvmnt[ty,ry,a,s,r]<0)stop("negative movement in reward tags!!!")
        }

        #Calculate probability of death by natural mortality and those that survive
        RewardCaptureProb[(fisheries+1)]=M[ry]/Z[ry,a,r]*(1-Survival[ry,a,r])
        RewardCaptureProb[(fisheries+2)]=Survival[ry,a,r]
        #Loop over fisheries
        for(f in 1:fisheries){
          #Calculate the capture probability for each fishery
          RewardCaptureProb[f]=F[ry,a,r,f]/Z[ry,a,r]*(1-Survival[ry,a,r])
        }
        #End fisheries loop
        #Determine tag fate using MULTINOMIAL distribution
        RewardTagFate[ty,ry,a,s,r]=rmultinom(n=1,size=RewardTagMvmnt[ty,ry,a,s,r],
        prob=RewardCaptureProb)
        #store the tags that are recaptured by fishery
        RewardTagsRecaptured[ty,ry,a,s,r]=RewardTagFate[ty,ry,a,s,r,1:fisheries]
        #test to make sure tags aren't created or destroyed during tag fate
        calculations
        if(sum(RewardTagFate[ty,ry,a,s,r,]) != RewardTagMvmnt[ty,ry,a,s,r]) stop("
        something not adding up in reward tag movement 1")
      }
      #End regions loop
      #test to make sure tags aren't created or destroyed anywhere
      if(sum(RewardTagFate[ty,ry,a,s,,]) != sum(RewardTagMvmnt[ty,ry,a,s,])) stop("
      something not adding up in movement 2 reward tags")
      #check to make sure that tags weren't created or destroyed
      if(round(RewardTagsAlive[ty,ry,a,s]*(TagRetention[(ry-ty+1),s])) != sum(
      RewardTagFate[ty,ry,a,s,,])){stop("Something does not add up in the reward tagging")}
      #Progress those fish that survive to the next year and age
      #remove those fish that die from the sample size of released fish i.e. only
      keep survivals
      if(a<(ages-1)){
        RewardTagsAlive[ty,(ry+1),(a+1),s]=sum(RewardTagFate[ty,ry,a,s,,(fisheries
        +2)])
      } else if(a==ages){
        RewardTagsAlive[ty,(ry+1),a,s]=sum(RewardTagFate[ty,ry,(ages-1),s,,(
        fisheries+2)]+RewardTagFate[ty,ry,ages,s,,(fisheries+2)])
      }
      #End if else for plus group calculations
    }
  }
}

```

```

    }
  }
}
#End stocks loop
#End age loop
#End capture year loop
#End tagging year loop

#Calculate the tag returns by summing over ages
RewardTagReturns=colSums(aperm(RewardTagsRecaptured, perm=c(3,1,2,4,5,6)), dim=1)

#reformat the Tag returns to get rid of the dimension for region of recapture
#This assumed that each fishery is only active in one region
#Also apply the reporting rate for that fishery
#This will only work if the fishery is active in only one region
RewardTagsReported=array(0, dim=c(years, years, stocks, fisheries))
RewardNeverRecovered=matrix(data = 0, nrow=years, ncol = stocks)
for(ty in 1:years){
  for(ry in ty:years){
    for(s in 1:stocks){
      for(f in 1:fisheries){
        tempr=which(FisheryActive[f,]==1)
        RewardTagsReported[ty, ry, s, f]=RewardTagReturns[ty, ry, s, tempr, f]
      }
    }
  }
}

for(y in 1:years){
  for(s in 1:stocks){
    RewardNeverRecovered[y, s]=RewardTagsReleased[y, s]-sum(RewardTagsReported[y, , s, ])
  }
} else {#End if statement for HighRewardType
RewardTagsReleased=matrix(0, nrow=years, ncol=stocks)
RewardTagsReported=array(0, dim=c(years, years, stocks, fisheries))
RewardReleaseAge=array(0, dim=c(years, ages, stocks))
RewardNeverRecovered=matrix(data = 0, nrow=years, ncol = stocks)
}

#####Calculate the parameters that need to be included in the data file for
comparison to parameter estimates

LastYearN=numeric(stocks)
for (s in 1:stocks) LastYearN[s]=sum(N[years, , s])

#####Create .dat file

#This puts in the first line description and creates the file or overwrites existing file since
append=false
cat(c("#Simulated data to be read into the assessment model using ADMB", "\n"), file="SimulatedData.
dat", append=FALSE)

#Prints a bunch of variables
cat(c("#number of years", years, "#number of regions", regions, "#number of stocks", stocks, "#number of
fisheries", fisheries, "#Number of age classes", ages, "#Phase of Natural Mortality estimation",
PhaseM, "#Phase of time-varying Natural Mortality", MVaryPhase, "#True Value of Natural Mortality"
, t(M)), sep="\n", append=TRUE, file="SimulatedData.dat")

#Print out the Type of Natural Mortality estimation that will be used 1== constant 2 == 5 year
block 3 == random walk
cat(c("", "#This is the M Estimation Type 1== constant 2 == 5 year block 3== random walk", MEst), file
="SimulatedData.dat", append=TRUE, sep="\n")

#Prints out Tag Loss
cat(c("#This is the Tag retention as a yearly percentage with tag for years after tagging", "\n"),
file="SimulatedData.dat", append = TRUE, sep=" ")
write.table(TagRetention, file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names
= FALSE)

```

```

#Prints out Reporting Rate info
cat(c("","#Phase of Reporting Rate estimated", PhaseRR, "#Phase of time-varying Reporting Rate",
      RRVaryPhase,"#This is the initial guess for the reporting rate parameters or value if not
      estimated", t(rr)), file="SimulatedData.dat", append = TRUE, sep="\n")

#Print out the Type of reporting Rate estimation that will be used 1== constant 2 == 5 year block
3 == random walk
cat(c("","#This is the RR Estimation Type 1== constant 2 == 5 year block 3== random walk", RREst),
    file="SimulatedData.dat", append=TRUE, sep="\n")

#Prints the True Mvmnt matrix
cat(c("\n", "#Matrix of True Movement parameters and used to calculate starting values", "\n"), file="
      SimulatedData.dat", sep="", append=TRUE)
write.table(Movement, file="SimulatedData.dat", sep=" ", append=TRUE, row.names=FALSE, col.names=FALSE)

#Prints fishery active matrix
cat(c("\n", "#Matrix of fishery active", "\n"), file="SimulatedData.dat", sep="", append=TRUE)
write.table(FisheryActive, file="SimulatedData.dat", sep=" ", append=TRUE, row.names = FALSE, col.names
            = FALSE)

#Prints observed Catch Data
cat(c("\n", "\n", "#The observed Catch data for the fisheries", "\n", "\n"), file="SimulatedData.dat",
    append=TRUE, sep="")
write.table(round(ObservedCatch), "SimulatedData.dat", sep=" ", append=TRUE, row.names = FALSE, col.
            names = FALSE)

#prints Fishery Effort Data
cat(c("\n", "#This is the observed Effort for the data", "\n", "\n"), file="SimulatedData.dat", append=
      TRUE, sep="")
write.table(round(ObservedEffort, 2), "SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.
            names = FALSE)

#Print True Fishery Catchability coefficient
cat(c("\n", "#This is the True fisheries Catchability coefficient parameter TrueQ", "\n \n"), file="
      SimulatedData.dat", append=TRUE, sep="")
write.table(q[1,], "SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names = FALSE)

#Print Survey Data
cat(c("\n #This is the observed Survey Data \n \n"), file="SimulatedData.dat", append=TRUE, sep="")
write.table(ObservedSurvey, "SimulatedData.dat", append=TRUE, sep=" ", row.names=FALSE, col.names =
            FALSE)

#Print True Survey catchability coefficient
cat(c("\n", "#This is the True Survey Catchability Coefficient parameter TrueSurveyQ", "\n \n"), file=
      "SimulatedData.dat", append=TRUE, sep="")
write.table(qSurvey, "SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE, col.names = FALSE)

#Prints Observed Age Composition
#ObservedAgeComp=aperm(ObservedAgeComp, perm=c(1,3,2))
cat(c("\n", "#This is the simulated age composition", "\n", "\n"), file="SimulatedData.dat", sep="",
    append=TRUE)
write.table(ObservedAgeComp, file="SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE, col.
            names = FALSE)

#Print out FisheryFullySelected age
cat("\n #This is the Age that is fully selected in the respective fishery to be used to set fully
      selected value \n \n", file="SimulatedData.dat", sep="", append=TRUE)
write.table(FisheryFullySelected, file = "SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE,
            col.names = FALSE)

#Prints out the True Fishery Selectivity Parameters
cat(c("\n", "#This is the True Selectivity Parameters excluding the fully selected", "\n", "\n"), file=
      "SimulatedData.dat", sep="", append=TRUE)
TrueSelectivity=matrix(NA, nrow=(ages-1), ncol=fisheries)
for (f in 1:fisheries){ TrueSelectivity[, f]=selectivity[-FisheryFullySelected[f], f]}
write.table(TrueSelectivity, file="SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE, col.
            names = FALSE)

#Print out Observed Survey Age Composition
cat(c("\n #This is the Observed Survey Age composition \n \n"), file="SimulatedData.dat", sep="",
    append=TRUE)
write.table(ObservedSurveyAgeComp, file="SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE,

```

```

col.names = FALSE)

#Print out SurveyFullySelected
cat("\n #This is the age that is fully selected in the survey to be used to the fully selected age
    in the assessment \n \n", file="SimulatedData.dat", sep="", append=TRUE)
write.table(SurveyFullySelected, file = "SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE,
    col.names = FALSE)

#Prints out True Survey Selectivity Parameters
cat(c("\n", "#This is the True Survey Selectivity Parameters excluding the fully selected", "\n", "\n"
    ), file="SimulatedData.dat", sep="", append=TRUE)
TrueSurveySel=matrix(NA, nrow=(ages-1), ncol=regions)
for (r in 1:regions){ TrueSurveySel[,r]=SurveySel[-SurveyFullySelected[r], r]}
write.table(TrueSurveySel, file="SimulatedData.dat", append=TRUE, sep = " ", row.names = FALSE, col.
    names = FALSE)

#Prints out the True Initial Abundance TrueN0
cat(c("\n", "#This is the True values of the initial Abundance TrueN0", "\n", "\n"), file="
    SimulatedData.dat", sep="", append=TRUE)
write.table(N[1,2:ages,], file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names
    = FALSE)

#Calculate and Print out True Mean Recruitment
LogMeanRecruitment=colMeans(log(N[1:years,1,]))
cat(c("\n", "#This is the True Mean Recruitment", "\n \n"), file="SimulatedData.dat", sep="", append=
    TRUE)
write.table(LogMeanRecruitment, file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.
    names = FALSE)

#Print out True Annual Recruitment
cat(c("\n", "#This is the True Annual Recruitment", "\n \n"), file="SimulatedData.dat", sep="", append=
    TRUE)
write.table(N[1:years,1,], file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names
    = FALSE)

#Print out the True Catch Sigma
cat(c("\n", "#This is the True Sigma Catch", "\n \n"), file="SimulatedData.dat", sep="", append=TRUE)
write.table(logcatchCV$sd, file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names
    = FALSE)

#Print out True Last Year's Abundance summed over ages
cat(c("\n #This is the True Last Years' Abundance \n \n "), file="SimulatedData.dat", sep="", append=
    TRUE)
write.table(LastYearN, file="SimulatedData.dat", append=TRUE, sep=" ", row.names = FALSE, col.names =
    FALSE)

#Print out test number 1
cat(c("\n", "#This is the first test number", "\n", 1234567890), file="SimulatedData.dat", append = TRUE
    , sep="")

#Print out reported tag returns
cat(c("\n", "\n", "#This is the Tags Reported", "\n", "\n"), file="SimulatedData.dat", append=TRUE, sep="")
write.table(aperm(TagsReported, perm=c(1,4,2,3)), file="SimulatedData.dat", append=TRUE, row.names =
    FALSE, col.names = FALSE, sep=" ")

#Print out the True Reporting Rate only if it is estimated
if(PhaseRR>0 | RRVaryPhase>0){
    cat(c("\n", "#This is the True Mean Reporting Rate", "\n", "\n"), file="SimulatedData.dat", append=
        TRUE, sep="")
    write.table(ReportingRate, file="SimulatedData.dat", append=TRUE, row.names = FALSE, col.names =
        FALSE, sep=" ")
}

#Print out test number 2
cat(c("", "#This is the second test number", 1234567890, ""), file="SimulatedData.dat", append = TRUE,
    sep="\n")

#Prints out Tags released by age
cat(c("#This is the Tags Released by Age, year and stock", ""), file="SimulatedData.dat", append =
    TRUE, sep="\n")
write.table(aperm(ReleaseAge, perm=c(1,3,2)), file="SimulatedData.dat", append=TRUE, sep = " ", row.names
    = FALSE, col.names = FALSE)

```

```

#Prints out Total Tags Released
cat(c("","#This is the Total Tags Released by year and stock",""), file="SimulatedData.dat", append =
  TRUE, sep="\n")
write.table(TagsReleased, file="SimulatedData.dat", append=TRUE, sep =" ", row.names = FALSE, col.names
  = FALSE)

#Print out test number 3
cat(c("#This is the third test number",1234567890), file="SimulatedData.dat", append = TRUE, sep="\n")

#Prints out Tags Never Recovered
cat(c("","#This is the number of tags that are never recovered for each release event",""), file="
  SimulatedData.dat", append = TRUE, sep="\n")
write.table(NeverRecovered, file="SimulatedData.dat", append=TRUE, sep =" ", row.names = FALSE, col.
  names = FALSE)

#Print out test number 4
cat(c("","#This is the fourth test number",1234567890), file="SimulatedData.dat", append = TRUE, sep="
  \n")

#Print out reported tag returns
cat(c("\n","\n","#This is the Reward Tags Reported","\n","\n"), file="SimulatedData.dat", append=TRUE
  , sep="\n")
write.table(aperm(RewardTagsReported, perm=c(1,4,2,3)), file="SimulatedData.dat", append=TRUE, row.
  names = FALSE, col.names = FALSE, sep=" ")

#Prints out Tags released by age
cat(c("#This is the Reward Tags Released by Age, year and stock",""), file="SimulatedData.dat",
  append = TRUE, sep="\n")
write.table(aperm(RewardReleaseAge, perm=c(1,3,2)), file="SimulatedData.dat", append=TRUE, sep =" ", row
  .names = FALSE, col.names = FALSE)

#Prints out Total Tags Released
cat(c("","#This is the Total Reward Tags Released by year and stock",""), file="SimulatedData.dat",
  append = TRUE, sep="\n")
write.table(RewardTagsReleased, file="SimulatedData.dat", append=TRUE, sep =" ", row.names = FALSE, col
  .names = FALSE)

#Prints out Tags Never Recovered
cat(c("","#This is the number of Reward tags that are never recovered for each release event",""),
  file="SimulatedData.dat", append = TRUE, sep="\n")
write.table(RewardNeverRecovered, file="SimulatedData.dat", append=TRUE, sep =" ", row.names = FALSE,
  col.names = FALSE)

#Print out test number 5
cat(c("","#This is the fifth test number",1234567890), file="SimulatedData.dat", append = TRUE, sep="
  \n")

#Print out the True abundance at age for each of the stocks. This won't be read into the admb file
  but it might be useful later
cat("\n #This is the True Abundance at Age for the stocks \n \n", file="SimulatedData.dat", append=
  TRUE)
write.table(aperm(N[1:years, ], perm=c(1,3,2)), file="SimulatedData.dat", append=TRUE, sep =" ", row.
  names = FALSE, col.names = FALSE)

#Print out the Fishing mortality with the random variation. Won't be read into the admb file but
  might be important later
cat("\n #This is the True Fishing Mortality with random variation \n \n", file="SimulatedData.dat",
  append=TRUE)
write.table(FFull, file="SimulatedData.dat", append=TRUE, sep =" ", row.names = FALSE, col.names =
  FALSE)

#Print out test number 6
cat(c("","#This is the sixth test number",1234567890), file="SimulatedData.dat", append = TRUE, sep="
  \n")

#####
#Code to write a stochastic starting value for caa.pin, release.pin and recapture.pin all at once
  with the same values in all three files (of shared parameters)

StartCV=.05

```



```

#cat("# Log Recruits \n", file="recapture.pin", append=FALSE)
cat("# Log Recruits \n", file="release.pin", append=FALSE)
#cat("# Log Recruits \n", file="caa.pin", append=FALSE)
StartLogRec=t(rnorm(n=length(LogMeanRecruitment), mean=LogMeanRecruitment, sd=abs(LogMeanRecruitment)
  *StartCV))
#write.table(StartLogRec, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
  FALSE)
write.table(StartLogRec, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
#write.table(StartLogRec, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# Log N0 \n", file="recapture.pin", append=TRUE)
cat("# Log N0 \n", file="release.pin", append=TRUE)
#cat("# Log N0 \n", file="caa.pin", append=TRUE)
StartLogN0=matrix(rnorm(length(N[1,1]), log(rowMeans(N[1,2:ages,])), abs(log(N[1,2:ages,]) *StartCV)),
  ,nrow=1, ncol=stocks)
#write.table(StartLogN0, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
  FALSE)
write.table(StartLogN0, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
#write.table(StartLogN0, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# Log N0 Devs\n", file="recapture.pin", append=TRUE)
#write.table(matrix(0, ncol=(ages-1), nrow=stocks), file="recapture.pin", sep=" ", append=TRUE, row.names
  = FALSE, col.names = FALSE)
cat("# Log N0 Devs\n", file="release.pin", append=TRUE)
write.table(matrix(0, ncol=(ages-1), nrow=stocks), file="release.pin", sep=" ", append=TRUE, row.names =
  FALSE, col.names = FALSE)
#cat("# Log N0 Devs\n", file="caa.pin", append=TRUE)
#write.table(matrix(0, ncol=(ages-1), nrow=stocks), file="caa.pin", sep=" ", append=TRUE, row.names =
  FALSE, col.names = FALSE)

#cat("# Log Q \n", file="recapture.pin", append=TRUE)
cat("# Log Q \n", file="release.pin", append=TRUE)
#cat("# Log Q \n", file="caa.pin", append=TRUE)
StartLogQ=t(rnorm(length(q[1,]), mean=log(q[1,]), sd=abs(log(q[1,]) *StartCV)))
#write.table(StartLogQ, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE
  )
write.table(StartLogQ, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
#write.table(StartLogQ, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# LogSurveyQ \n", file="recapture.pin", append=TRUE)
cat("# LogSurveyQ \n", file="release.pin", append=TRUE)
#cat("# LogSurveyQ \n", file="caa.pin", append=TRUE)
StartLogSrvyQ=t(rnorm(length(qSurvey), log(qSurvey), abs(log(qSurvey) *StartCV)))
#write.table(StartLogSrvyQ, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
  FALSE)
write.table(StartLogSrvyQ, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
  FALSE)
#write.table(StartLogSrvyQ, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# slctvty \n", file="recapture.pin", append=TRUE)
cat("# slctvty \n", file="release.pin", append=TRUE)
#cat("# slctvty \n", file="caa.pin", append=TRUE)
StartSlctvty=matrix(rnorm(length(TrueSelectivity), TrueSelectivity, abs(TrueSelectivity *StartCV)),
  ,nrow=(ages-1))
StartSlctvty[StartSlctvty <= 0] = 0.001
StartSlctvty[StartSlctvty >= 5] = 4.99
StartSlctvty[is.nan(StartSlctvty)]=1
#write.table(StartSlctvty, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
  FALSE)
write.table(StartSlctvty, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE
  )
#write.table(StartSlctvty, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

StartSrvySlctvty=matrix(rnorm(length(TrueSurveySel), TrueSurveySel, abs(TrueSurveySel *StartCV)), nrow
  =(ages-1))
StartSrvySlctvty[StartSrvySlctvty <= 0] = 0.001
StartSrvySlctvty[StartSrvySlctvty >= 5] = 4.99
StartSrvySlctvty[is.nan(StartSrvySlctvty)]=1
#cat("# SrvySlctvty \n", file="recapture.pin", append=TRUE)
#write.table(StartSrvySlctvty, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names

```

```

= FALSE)
cat("# SrvySltvty \n", file="release.pin", append=TRUE)
write.table(StartSrvySltvty, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
#cat("# SrvySltvty \n", file="caa.pin", append=TRUE)
#write.table(StartSrvySltvty, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)

#cat("# LogRecruitmentDevs \n", file="recapture.pin", append=TRUE)
#write.table(matrix(0, nrow=(years-3), ncol=stocks), file="recapture.pin", sep=" ", append=TRUE, row.
names = FALSE, col.names = FALSE)
cat("# LogRecruitmentDev1 \n", file="release.pin", append=TRUE)
write.table(matrix(0, nrow=(years-3), ncol=stocks), file="release.pin", sep=" ", append=TRUE, row.names =
FALSE, col.names = FALSE)
#cat("# LogRecruitmentDev1 \n", file="caa.pin", append=TRUE)
#write.table(matrix(0, nrow=(years-3), ncol=stocks), file="caa.pin", sep=" ", append=TRUE, row.names =
FALSE, col.names = FALSE)

#cat("# LogEffortDevs \n", file="recapture.pin", append=TRUE)
#write.table(matrix(0, nrow=(years-1), ncol=fisheries), file="recapture.pin", sep=" ", append=TRUE, row.
names = FALSE, col.names = FALSE)
cat("# LogEffortDev1 \n", file="release.pin", append=TRUE)
write.table(matrix(0, nrow=(years-1), ncol=fisheries), file="release.pin", sep=" ", append=TRUE, row.
names = FALSE, col.names = FALSE)
#cat("# LogEffortDev1 \n", file="caa.pin", append=TRUE)
#write.table(matrix(0, nrow=(years-1), ncol=fisheries), file="caa.pin", sep=" ", append=TRUE, row.names =
FALSE, col.names = FALSE)

StartLogCatchCV=rnorm(regions, log(logcatchCV$sd), abs(log(logcatchCV$sd)*StartCV))
#cat("# LogSigmaCatch \n", file="recapture.pin", append=TRUE)
#write.table(StartLogCatchCV, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
cat("# LogSigmaCatch \n", file="release.pin", append=TRUE)
write.table(StartLogCatchCV, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
#cat("# LogSigmaCatch \n", file="caa.pin", append=TRUE)
#write.table(StartLogCatchCV, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE
)

#cat("# Mvmnt \n", file="recapture.pin", append=TRUE)
cat("# Mvmnt \n", file="release.pin", append=TRUE)
#cat("# Mvmnt \n", file="caa.pin", append=TRUE)
StartMvmnt=matrix(rnorm(length(Movement[, -4]), log(Movement[, -4]/(1-rowSums(Movement[, -4]))), abs(log
(Movement[, -4]/(1-rowSums(Movement[, -4])))*StartCV)), nrow=4)
StartMvmnt[StartMvmnt <= -6] = -6
StartMvmnt[StartMvmnt >= 6] = 6
StartMvmnt[is.nan(StartMvmnt)]=rnorm(length(StartMvmnt[is.nan(StartMvmnt)]), 0, 1)
#write.table(StartMvmnt, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
write.table(StartMvmnt, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
#write.table(StartMvmnt, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# RR \n", file="recapture.pin", append=TRUE)
cat("# RR \n", file="release.pin", append=TRUE)
StartRR=rnorm(length(rr), -log((1/rr)-1), abs(-log((1/rr)-1)*StartCV))
StartRR[StartRR <= -6] = -6
StartRR[StartRR >= 6] = 6
StartRR[is.nan(StartRR)]=6
#write.table(t(StartRR), file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
write.table(t(StartRR), file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)

#cat("# LogM \n", file="recapture.pin", append=TRUE)
cat("# LogM \n", file="release.pin", append=TRUE)
#cat("# LogM \n", file="caa.pin", append=TRUE)
StartLogM=rnorm(1, log(M[1]), abs(log(M[1])*StartCV))
if (PhaseM<0)
{
# write.table(0, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
write.table(0, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
# write.table(0, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
} else{

```



```

StartLogM=rnorm(1,log(M[1]),abs(log(M[1])*StartCV))
# write.table(StartLogM, file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
write.table(StartLogM, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =
FALSE)
# write.table(StartLogM, file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
}

if (RREst==1)
{
  cat("# LogRRDevs \n", file="release.pin", append=TRUE)
  write.table(matrix(0, ncol=fisheries, nrow=1), file="release.pin", sep=" ", append=TRUE, row.names
= FALSE, col.names = FALSE)
}

if (RREst==2)
{
  cat("# LogRRDevs \n", file="release.pin", append=TRUE)
  StartRRTV=rnorm(length(ReportingRate[c(1,6,11,16,21,26,31,36)]), -log((1/ReportingRate[c
(1,6,11,16,21,26,31,36)])-1), abs(-log((1/ReportingRate[c(1,6,11,16,21,26,31,36)])-1)*StartCV)
)
  StartRRTV[StartRRTV <= -6] = -6
  StartRRTV[StartRRTV >= 6] = 6
  StartRRTV[is.nan(StartRRTV)]=6
  write.table(matrix(StartRRTV, ncol=fisheries, nrow=(years/5)), file="release.pin", sep=" ", append=
TRUE, row.names = FALSE, col.names = FALSE)
}

if (RREst==3)
{
  cat("# LogRRDevs \n", file="release.pin", append=TRUE)
  write.table(matrix(0, ncol=fisheries, nrow=(years-1)), file="release.pin", sep=" ", append=TRUE, row.
names = FALSE, col.names = FALSE)
}

if (MEst ==1)
{
  cat("# LogMDevs \n", file="release.pin", append=TRUE)
  write.table(0, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
}

if (MEst ==2)
{
  cat("# LogMDevs \n", file="release.pin", append=TRUE)
  write.table(t(rnorm(8, log(M[c(1,6,11,16,21,26,31,36)]), abs(log(M[c(1,6,11,16,21,26,31,36)])) *
StartCV))), file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names = FALSE)
}

if (MEst ==3)
{
  cat("# LogMDevs \n", file="release.pin", append=TRUE)
  write.table(t(rnorm(years, log(M), abs(log(M)*StartCV))), file="release.pin", sep=" ", append=TRUE,
row.names = FALSE, col.names = FALSE)
}

```

ITCAAN Model Code

```

TOP_OF_MAIN_SECTION
  armblsize = 2000000000; // use instead of gradient_structure::set_ARRAY_MEMBLOCK_SIZE
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(10000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(50000000);

GLOBALS_SECTION
#include <admodel.h>
#include <qfclib.h>

//From Vandergoot walleye movement code
//This function calculates the movement rate using a parameter for all but the last region and
//converts to logit scale so the values are between 0 and 1
dvar_vector LogitProp(const dvar_vector& a)
{
  int dim;
  dim=a.size()+1;
  dvar_vector p(1,dim);
  dvar_vector expa=exp(a);
  p(1,dim-1)=expa/(1.+sum(expa));
  //p(dim)=1.-sum(p(1,dim-1));
  p(dim)=1./(1.+sum(expa));
  return p;
}

DATA_SECTION
//change the name of the file that will contain the simulated data
!! ad.comm::change_datafile_name("SimulatedData.dat");

init_int years //number of years
init_int regions //number of regions
init_int stocks //number of stocks
init_int fisheries //number of fisheries
init_int ages //number of ages modeled
//Variables that are not read in. creates variables from read in ones
int yearsp1 //years plus 1
int yearsm1 //years minus 1
int yearsm2 //years minus 2
int yearsby5 //Number of 5 year blocks in time series
int agesm1 //ages minus 1
int regionsm1 //Number of regions minus 1

LOCAL_CALCULATIONS
//Calculate variables to be used to create some parameter vectors
yearsp1=years+1;
yearsm1=years-1;
yearsm2=years-2;
agesm1=ages-1;
yearsby5=years/5;
regionsm1=regions-1;
END_CALCULATIONS

//More read in data
init_int PhaseM //variable whether to estimate M or not. If it is negative do not
estimate if positive it is estimated in that phase
init_int MVaryPhase //Variable for if a time varying M is estimated or not. If it
is negative do not estimate if positive it is estimated in that phase
init_vector TrueM(1,years) //natural mortality value if phaseM is positive
then you need to transform this starting value so it is on the logistic scale
init_number MEst //Variable to determine which type of natural mortality estimation
is going to be used 1==constant 2==5 year block 3==random walk
int Mlength
LOCAL_CALCULATIONS
if (MEst==2){Mlength=yearsby5;
} else if (MEst ==3){ Mlength=years;
} else{ Mlength=1; }
END_CALCULATIONS

init_matrix TagRetention(1,years,1,stocks) //Tag loss of tagged fish will be a
percentage lost annually each year
init_number PhaseRR //Variable for if a reporting rate is estimated or not. If it is
negative do not estimate if positive it is estimated in that phase

```

```

init_number RRVaryPhase          //Variable for if a time-varying reporting rate is estimated
or not. If it is negative do not estimate if positive it is estimated in that phase
init_vector rr(1,fisheries)      //Initial starting value for the Reporting rate or the
value of the parameter if not estimated
init_number RREst                //Variable to determine which type of reporting rate estimation is
going to be used 1==constant 2==5 year block 3==random walk
int RRLength
LOCALCALCS
  if(RREst==2){RRLength=yearsby5;
  }else if (RREst ==3){ RRLength=yearsml;
  }else{ RRLength=1;      }
ENDCALCS
init_matrix TrueMvmnt(1,regions,1,regions) //Matrix of the starting values to set for the
Mvmnt. On the logit scale and calculates the last regions
init_matrix FisheryActive(1,fisheries,1,regions) //Indicator variable for if fisheries are
active in a region
init_matrix ObservedCatch(1,years,1,fisheries) //Observed total Catch by fisheries
init_matrix ObservedEffort(1,years,1,fisheries) //Observed fishing effort by fishery
init_vector TrueQ(1,fisheries) //True Fishery Catchability Coefficient parameters
init_matrix ObservedSurvey(1,years,1,regions) //Observed Catch Per Unit Effort from each
region by a survey
init_vector TrueSurveyQ(1,regions) //True Catchability coefficient for the surveys
parameters
init_3darray ObservedAgeComp(1,years,1,fisheries,1,ages) //Observed age composition by fishery
init_vector FisheryFullySelected(1,fisheries) //The age that is fully selected for each
fishery
init_matrix TrueSel(1,agesml,1,fisheries) //True selectivity parameter matrix
init_3darray ObservedSurveyAgeComp(1,years,1,regions,1,ages) //Observed Age Composition from the
survey for each region
init_vector SurveyFullySelected(1,regions) //The age that is fully selected to the
survey in each region
init_matrix TrueSurveySel(1,agesml,1,regions) //True Selectivity Parameters for the
survey from each region
init_matrix TrueN0(2,ages,1,stocks) //True Initial Abundance parameters
init_vector TrueMeanRecruits(1,stocks) //True Mean Recruitment parameters
init_matrix TrueRecruits(1,years,1,stocks) //True Annual Recruitment parameters
init_number TrueSigmaCatch //True Catch Sigma to compare to LogSigmaCatch
init_vector TrueLastYearN(1,stocks) //True Abundance summed over ages for all stocks
init_number test1 //test value
// test to see if age composition has been read in correctly
!!if (test1 != 1234567890){cout << "Test 1 not read correctly" << endl; exit(10);}
init_4darray TagsReported(1,years,1,stocks,1,years,1,fisheries) //Tags Reported for release year,
recapture years, release stock, fishery of recapture
!!if (PhaseRR>0 | RRVaryPhase>0)
  init_matrix TrueRR(1,years,1,fisheries)

init_number test2 //Test value 2
// test to see if the tag returns have been read in correctly
!!if(test2 != 1234567890){cout << "Test 2 not read correctly" << endl; exit(11);}
init_3darray ReleaseAge(1,years,1,ages,1,stocks) //Number of Tags released by age for
calculations
init_matrix TagsReleased(1,years,1,stocks) //Total number of tags released by year and stock
init_number test3 //Test value 3
// test to see if the tags released have been read in correctly
!!if(test3 != 1234567890){cout << "Test 3 not read correctly" << endl; exit(12);}
init_matrix NeverRecovered(1,years,1,stocks) //Number of tags that are never recovered in
simulated data
init_number test4 //Test value 4
// test to see if the tags never returned have been read in correctly
!!if(test4 != 1234567890){cout << "Test 4 not read correctly" << endl; exit(13);}
init_4darray RewardTagsReported(1,years,1,stocks,1,years,1,fisheries) //Tags Reported for release
year, recapture years, release stock, fishery of recapture
init_3darray RewardReleaseAge(1,years,1,ages,1,stocks) //Number of Tags released by age for
calculations
init_matrix RewardTagsReleased(1,years,1,stocks) //Total number of tags released by year and
stock
init_matrix RewardNeverRecovered(1,years,1,stocks) //Number of tags that are never recovered in
simulated data
init_number test5 //Test value 5
// test to see if the tags never returned have been read in correctly
!!if(test5 != 1234567890){cout << "Test 5 not read correctly" << endl; exit(14);}
init_3darray NTrue(1,years,1,ages,1,stocks) // The True abundance at age from the simulation

```

```

init_matrix TrueF(1,years,1,fisheries)
init_number test6 //Test value 6
// test to see if the tags never returned have been read in correctly
!!if(test6 != 1234567890){cout << "Test 6 not read correctly" << endl; exit(15);}

int y //indice to keep track of years
int s //indice to keep track of stock
int r //indice to keep track of region
int r2 //indice to keep track of second region for movement calculations
int f //indice to keep track of fishery
int a //indice to keep track of age
int ty //indice to keep track of tagging year
int ry //indice to keep track of recapture year
// !!cout<<"Finished Data Section"<<endl;

```

PARAMETER_SECTION

```

//Parameters to estimate
init_bounded_vector LogRecruits(1,stocks,5.,25.,1) //Log of mean recruitment for each stock
init_bounded_vector LogN0_mean(1,stocks,5.,25.,1)
init_bounded_dev_vector N01(2,ages,-10,10,5)
init_bounded_dev_vector N02(2,ages,-10,10,5)
init_bounded_dev_vector N03(2,ages,-10,10,5)
init_bounded_dev_vector N04(2,ages,-10,10,5)
init_bounded_vector LogQ(1,fisheries,-20.,-2.,1) //Catchability coefficient for fisheries
init_bounded_vector LogSurveyQ(1,regions,-20.,-2.,1) //Catchability coefficient for surveys
init_bounded_matrix slctvty(1,agesm1,1,fisheries,0.,5.,1) //Selectivity parameters without the
fully selected age
init_bounded_matrix SrvySlctvty(1,agesm1,1,regions,0.,5.,1) //Selectivity parameters for the
survey without the fully selected ages which varies by region
init_bounded_matrix LogRecruitmentDevs(2,yearsm2,1,stocks,-10.,10.,3) //Recruitment deviation
vector for stock 1 will be put into matrix for calculations
init_bounded_matrix LogEffortDevs(2,years,1,fisheries,-5.,5.,2) //Catchability Coefficient
deviation vector for stock 4 will be put into matrix for calculations
init_bounded_vector LogSigmaCatch(1,regions,-6.,2.,6) //Log SD for catch
init_bounded_matrix Mvmnt(1,stocks,1,regionsm1,-6.,6.,1) //Movement parameters for all but
last region will be converted to logit scale
init_bounded_vector RR(1,fisheries,-10,10,PhaseRR) //Reporting Rate for each fishery will
be converted to logit scale
init_bounded_number LogM(-10,1,PhaseM) //Natural Mortality estimated value
init_bounded_matrix LogRRDevs(1,RRlength,1,fisheries,-10,10,RRVaryPhase) //Deviations for annual
reporting rate for each year
init_bounded_vector LogMDevs(1,Mlength,-10,1,MVaryPhase) //Natural Mortality deviation
vector to calculate time-varying M
objective_function_value nll //Objective negative log likelihood value
//Variables that are calculated from the estimated parameters
matrix Selectivity(1,ages,1,fisheries) //All Selectivity Parameters for the fisheries
matrix SurveySelectivity(1,ages,1,regions) //All Selectivity Parameters for the surveys
3darray N(1,yearsp1,1,ages,1,stocks) //Abundance of individuals by age and stock
matrix NAll(1,years,1,stocks) //Total Abundance by year and stock
matrix NAllTrue(1,years,1,stocks) //True Total Abundance of simulated stocks
vector NTotal(1,years) //Total Abundance by year and stock
vector NTotalTrue(1,years) //True Total Abundance of simulated stocks
4darray NMvmnt(1,years,1,ages,1,stocks,1,regions) //Abundance of fish after movement and
mortality in each region
matrix Movement(1,stocks,1,regions) //Rate of movement between regions calculated from
parameters
matrix Q(1,years,1,fisheries) //matrix of log catchability deviations
4darray F(1,years,1,fisheries,1,ages,1,regions) //Fishing mortality calculated from
catchability, effort, and selectivity
3darray FTotal(1,years,1,ages,1,regions) //Total Fishing mortality in a region summing
over fisheries
3darray CatchAge(1,years,1,fisheries,1,ages) //Number of fish caught in year by fisheries
and age
matrix TotalCatch(1,years,1,fisheries) //Total number of fish caught in a year by a
fishery
3darray Z(1,years,1,ages,1,regions) //Total Mortality in a region (Z=F+M)
3darray Survival(1,years,1,ages,1,regions) //Survival in a region calculated from total
mortality
3darray Deaths(1,years,1,ages,1,regions) //Deaths in a region calculation from 1-survival
4darray Baranov(1,years,1,fisheries,1,ages,1,regions) //matrix to store calculations of M/Z
*(1-Survival) to be used in catch calculation and tag returns
3darray AgeComp(1,years,1,fisheries,1,ages) //Proportions of age group in catch calculated

```

```

from CatchAge
3darray SurveyAgeComp(1,years,1,regions,1,ages) //Proportion of age group caught by each
survey
matrix SurveyQMatrix(1,ages,1,regions) //matrix to be filled with the estimated parameter
to be used in survey calculations
3darray SurveyMortality(1,years,1,ages,1,regions) //Array to store the calculation for the
combination of survey selectivity, catchability and mortality in the year up to occurrence (
October)
matrix TotalSurvey(1,years,1,regions) //matrix of the predicted survey CPUE for each
year and region
3darray SurveyAge(1,years,1,regions,1,ages) //Survey by age to be used to calculate
proportions and totals
number CatchNLL //negative log likelihood from catch
number EffortNLL //negative log likelihood from catchability coefficient deviations
number AgeCompNLL //negative log likelihood from age composition
number SurveyNLL //negative log likelihood from the surveys
number SurveyAgeCompNLL //negative log likelihood from the survey age composition
number TagNLL //negative log likelihood from tagging
number RewardTagNLL //negative log likelihood from high Rewardtagging
// Use variance ratio to calculate LogSigmaEffort in objective function from estimate of
LogSigmaCatch
number EffortVarianceRatio //Variance Ratio of the effort variance compared to the
catch variance
number SurveyVarianceRatio //Variance Ratio of the survey compared to the catch
variance
vector LogSigmaEffort(1,regions) //SD of catchability coefficient deviations for
likelihood calculations
vector LogSigmaSurvey(1,regions) //SD of error in the survey data
number LogSigmaRec //SD of error in Recruitment Deviations used to weight likelihood
number LogSigmaAbun //SD of error in initial abundance
number LogSigmaM //SD of error in Natural Mortality deviations to weight random walk
number LogSigmaRR //SD of error in Reporting Rate deviations to weight random walk
number RecruitmentNLL //negative log likelihood from recruitment deviations
number InitAbunNLL //negative log likelihood for initial abundance deviations
vector M(1,years) //vector for natural mortality
vector Mtemp(1,years) //temporary vector for untransformed natural mortality
4darray TagsAlive(1,years,1,years,1,ages,1,stocks) //Number of Tags alive at the beginning of
year (year of tag release, year of tag recapture/alive, age of fish,stock of fish release)
4darray RewardTagsAlive(1,years,1,years,1,ages,1,stocks) //Number of Reward Tags alive at
the beginning of year (year of tag release, year of tag recapture/alive, age of fish,stock of
fish release)
matrix TempNMvmnt(1,stocks,1,regions) //Temporary number to not repeat the
calculation of multiplying N and movement
5darray TagMvmnt(1,years,1,stocks,1,years,1,ages,1,regions) //Number of Tags that move to each
region (year of tag release, year of tag recapture/alive, age of fish,stock of fish release,
region of )
5darray RewardTagMvmnt(1,years,1,stocks,1,years,1,ages,1,regions) //Number of Reward Tags that
move to each region (year of tag release, year of tag recapture/alive, age of fish,stock of
fish release, region of )
4darray TagsCaught(1,years,1,stocks,1,years,1,fisheries) //Fate of tagged fish. (year of tag
releas, year of tag recapture, age of fish,stock of fish release, First f are captured by
fisheries)
4darray RewardTagsCaught(1,years,1,stocks,1,years,1,fisheries) //Fate of high reward tagged fish.
(year of tag releas, year of tag recapture, age of fish,stock of fish release, First f are
captured by fisheries)
matrix ReportingRate(1,years,1,fisheries) //The reporting rate for each year and fishery
value will be between 0 and 1
matrix RRtemp(1,years,1,fisheries) //Temporary matrix to calculate the random walk to
convert to Reporting Rate when RREst==3
4darray TagReturns(1,years,1,stocks,1,years,1,fisheries) //Tags Returned by year and fishery
they are summed over regions and ages
4darray RewardTagReturns(1,years,1,stocks,1,years,1,fisheries) //Tags Returned by year and
fishery they are summed over regions and ages
matrix TotalReturned(1,years,1,stocks) //Total number of tags returned for each release
matrix NotReturned(1,years,1,stocks) //Number of Tags that were never Recovered either not
caught, shed or not reported
matrix RewardTotalReturned(1,years,1,stocks) //Total number of tags returned for each
release
matrix RewardNotReturned(1,years,1,stocks) //Number of Tags that were never Recovered
either not caught, shed or not reported

vector LastYearN(1,stocks) //vector of the sum of abundance over ages for the last year for

```

```

    report
    vector zerovec(2,yearsm2);
    vector zerovec2(2,years);
    vector zerovec3(2,ages);
    vector maxSel(1,fisheries);
    vector maxSurveySel(1,regions);
    // !!cout<<"Finished Parameter Section"<<endl;

PRELIMINARY_CALCS_SECTION
//Set the starting values for various parameters
if (PhaseM<0 && MVaryPhase<0){
    M=TrueM;
}
if (PhaseRR<0 && RRVaryPhase<0){
    for (y=1;y<=years;y++)
        ReportingRate[y]=rr;
}
EffortVarianceRatio=1.;
SurveyVarianceRatio=0.2537;
LogSigmaRec=log(4.0);
LogSigmaAbun=log(4.0);
LogSigmaRR=log(2);
LogSigmaM=log(2);
// cout<<"Finished Preliminary Calcs"<<endl;

PROCEDURE_SECTION
CalculateParameters();
CalculateFZ();
CalculateN();
CalculateTagReturns();
CalculateObjectiveFunction();

FUNCTION CalculateParameters
//Initialize the parameters that will be calculated by this function
//Use logit function to calculate what the movement proportions will be
Movement.initialize();
for (s=1;s<=stocks;s++)
{
    Movement(s)=LogitProp(Mvmnt(s));
}
//insert the parameter estimates into the correct location in the selectivity matrices using the
known fully selected age
for (a=1;a<=ages;a++)
{
    for (r=1;r<=regions;r++)
    {
        if (a<SurveyFullySelected[r])
        {
            SurveySelectivity(a,r)=SrvySlctvty(a,r);
        }
        else if (a==SurveyFullySelected[r])
        {
            SurveySelectivity(a,r)=1;
        }
        else
        {
            SurveySelectivity(a,r)=SrvySlctvty((a-1),r);
        }
    }
    for (f=1;f<=fisheries;f++)
    {
        if (a<FisheryFullySelected[f])
        {
            Selectivity(a,f)=slctvty(a,f);
        }
        else if (a==FisheryFullySelected[f])
        {
            Selectivity(a,f)=1;
        }
        else
        {
            Selectivity(a,f)=slctvty((a-1),f);
        }
    }
}

```

```

    }
}
}
if (PhaseRR>0 || RRVaryPhase>0)
{
    //If Reporting Rate is estimated
    if (RREst==1)
    {
        //Reporting Rate is estimated but not time-varying
        for (y=1;y<=years;y++)
            ReportingRate[y]=1./(1.+exp(-RR));
    } else if (RREst==2)
    {
        //Reporting Rate is estimated in 5 year blocks
        for (y=1;y<=yearsby5;y++)
        {
            for (int temps=1;temps<=5;temps++)
            {
                ReportingRate[(y-1)*5+temps]=1./(1.+exp(-(LogRRDevs[y])));
            }
        }
    } else if (RREst==3)
    {
        //If Reporting Rate is estimated time-varying as a random walk
        ReportingRate[1]=1./(1.+exp(-RR));
        RRtemp[1]=RR;
        for (y=1;y<=yearsml;y++)
        {
            RRtemp[y+1]=RRtemp[y]+LogRRDevs[y];
            ReportingRate[y+1]=1./(1.+exp(-RRtemp[y+1]));
        }
    } else
    {
        cout<<"You must specify RREst equal to 1, 2 or 3"<<endl;
        exit(21);
    }
}
//If not estimated is already done in preliminary calcs and does not change
if (PhaseM>0 || MVaryPhase>0)
{
    //Natural Mortality estimated
    if (MEst==1)
    {
        M=exp(LogM); //Natural mortality is estimated constant
    } else if (MEst==2)
    {
        //Natural Mortality is estimated in 5 year blocks
        for (y=1;y<=yearsby5;y++)
        {
            for (int temps=1;temps<=5;temps++)
            {
                M[(y-1)*5+temps]=exp(LogMDevs[y]);
            }
        }
    } else if (MEst==3)
    {
        //Natural mortality is estimated as a Random walk
        M=exp(LogMDevs);
    } else
    {
        cout<<"You must specify MEst equal to 1, 2 or 3"<<endl;
        exit(31);
    }
}
//If not estimated is already done in preliminary calcs and does not change
// Fill in the Survey Q matrix to allow for elementwise calculations
for (a=1;a<=ages;a++)
{
    SurveyQMatrix[a]=mfexp(LogSurveyQ);
}
Q[1]=exp(LogQ);
for (y=2;y<=years;y++)

```



```

{
    Q[y]=elem_prod(Q[y-1],exp(LogEffortDevs[y]));
}
// cout<<"Finished Calculate Parameters"<<endl;

FUNCTION CalculateFZ
FTotal.initialize(); F.initialize(); Z.initialize(); Survival.initialize();
for (y=1;y<=years;y++)
{
    //Begin year loop
    for (a=1;a<=ages;a++)
    {
        //Begin age loop
        for (f=1;f<=fisheries;f++)
        {
            //Begin fisheries loop
            //Calculate fishery mortality from parameters
            F[y][f][a]=Q(y,f)*ObservedEffort(y,f)*Selectivity(a,f)*FisheryActive[f];
            for (r=1;r<=regions;r++)
            {
                //Begin region loop
                //Calculate total fishing mortality by summing over fisheries
                FTotal(y,a,r)+=F(y,f,a,r);
            } //End regions loop
        } //End fishery loop
    } //End ages loop
    //Calculate Total mortality
    Z[y]=FTotal[y]+M[y];
} //End year loop
// Calculate Survival
Survival = mfexp(-1.0*Z);
Deaths = 1-Survival;
for (y=1;y<=years;y++)
{
    //Begin year loop
    for (f=1;f<=fisheries;f++)
    {
        // Calculate F/Z *(1-Survival) to be used for catch at age and tagging
        Baranov[y][f]= elem_prod(elem_div(F[y][f],Z[y]),Deaths[y]);
    }
    // Calculate the mortality, catchability and selectivity that occur for each survey assume it
    // occurs in october so 10/12 is approximately 0.833333
    SurveyMortality[y]= elem_prod(elem_prod(mfexp(-0.833333333*Z[y]),SurveySelectivity),
    SurveyQMatrix);
}
// cout<<"Finished FZ"<<endl;

FUNCTION CalculateN
//Initialize variables used in this section
N.initialize(); NMvmnt.initialize(); CatchAge.initialize(); TotalCatch.initialize(); AgeComp.
initialize(); TotalSurvey.initialize(); SurveyAge.initialize(); TempNMvmnt.initialize();
//Initialize abundance calculated from estimated parameters
for (a=2;a<=ages;a++)
{
    N[1][a][1]=exp(LogN0_mean(1)+N01(a));
    N[1][a][2]=exp(LogN0_mean(2)+N02(a));
    N[1][a][3]=exp(LogN0_mean(3)+N03(a));
    N[1][a][4]=exp(LogN0_mean(4)+N04(a));
}
N[1][1]=exp(LogRecruits);
for (y=2;y<=(years-2);y++)
{
    N[y][1]=elem_prod(N[y-1][1],exp(LogRecruitmentDevs[y]));
}
// Recruitment of last 2 years is equal to average of 3 previous years
N[years-1][1]=(N[years-2][1]+N[years-3][1]+N[years-4][1])/3.0;
N[years][1]=(N[years-2][1]+N[years-3][1]+N[years-4][1])/3.0;
for (y=1;y<=years;y++)
{
    //Begin year loop
    for (a=1;a<=ages;a++)
    {
        //Begin age loop
        for (s=1;s<=stocks;s++)
        {
            //Begin stock loop
            //Calculate a row vector of fish that move to all the regions from one stock
            TempNMvmnt[s]=N(y,a,s)*Movement[s];
            //Calculate the area specific mortality for the fish in each region
            NMvmnt[y][a][s]=elem_prod(TempNMvmnt[s],Survival[y][a]);
        }
    }
}

```



```

        for (f=1;f<=fisheries;f++)
        { //Begin fishery loop
            //Calculate the catch for each area summing over the different spawning stocks
            CatchAge(y,f,a)+=sum(elem_prod(Baranov[y][f][a],TempNMvmnt[s]));
        } //End fishery loop
        //Calculate the Abundance at the next time step by summing survival over regions.
        Assumed a plus group calculation
        if((a<ages))
        {
            N((y+1),(a+1),s)=sum(NMvmnt[y][a][s]);
        }
        else{
            N((y+1),ages,s)+=sum(NMvmnt[y][ages][s]);
        } //End if/else ages
    } //End stock loop
    for (r=1;r<=regions;r++)
    { //Begin region loop
        SurveyAge[y][r][a]=sum(column(TempNMvmnt,r)*SurveyMortality(y,a,r));
    } //End region loop
    } //End ages loop
    for(f=1;f<=fisheries;f++)
    { //Begin fisheries loop
        // Calculate the Total Catch and proportion in each age class in the catch
        TotalCatch(y,f)=sum(CatchAge[y][f]);
        AgeComp[y][f]=CatchAge[y][f]/TotalCatch(y,f);
    } //End fisheries loop
    for(r=1;r<=regions;r++)
    { //Begin region loop
        // Calculate the total Survey and the proportion in each age class of the fish caught
        TotalSurvey(y,r)=sum(SurveyAge[y][r]);
        SurveyAgeComp[y][r]=SurveyAge[y][r]/TotalSurvey(y,r);
    } //End region loop
    } //End year loop
    // cout<<"Finished N"<<endl;

```

FUNCTION CalculateTagReturns

```

// This keeps track of releases by age, year and region of release for one release event and then
// which ones are recovered
TagsAlive.initialize(); TagMvmnt.initialize(); TotalReturned.initialize(); TagReturns.initialize
(); NotReturned.initialize(); RewardTagsAlive.initialize(); RewardTagMvmnt.initialize();
RewardTotalReturned.initialize(); RewardTagReturns.initialize(); RewardNotReturned.initialize()
;
for (ty=1;ty<(years-ages);ty++)
{ //Loop over tag release years
    //Don't loop over the last ages of years so not exceeding the bounds of the arrays. Will run
    // another loop for the remaining years
    // Initialize the Tags Alive as the number of tags released
    TagsAlive[ty][ty]=ReleaseAge[ty];
    RewardTagsAlive[ty][ty]=RewardReleaseAge[ty];
    for (s=1;s<=stocks;s++)
    { //Loop over stock of release
        if (TagsReleased(ty,s)>0)
        { //If there are actually tags released do calculations
            for (ry=ty;ry<(ty+ages);ry++)
            { //Loop over recapture years 1 starting from tag year and going only to the age where all
            // ages are in the plus group so don't need to do all of these calculations. Will run another loop
            // for just the plus group
                for (a=1;a<=ages;a++)
                { //Loop over Ages
                    //Calculate the tags that move to each region after applying a tag shedding rate
                    TagMvmnt[ty][s][ry][a]=TagsAlive(ty,ry,a,s)*(TagRetention((ry-ty+1),s))*Movement[s];
                    if (a<ages)
                    { //Begin If loop for ages
                        //Calculate the fish that are alive at the beginning of the next year
                        TagsAlive(ty,(ry+1),(a+1),s)=sum(elem_prod(TagMvmnt[ty][s][ry][a],Survival[ry][a]));
                    }
                    else
                    { //Continue If statement
                        //Calculate the fish that are alive at the beginning of the next year in
                        // the plus group
                        TagsAlive(ty,(ry+1),ages,s)+=sum(elem_prod(TagMvmnt[ty][s][ry][ages],
                        Survival[ry][ages]));
                    }
                }
            }
        }
    }
}

```

```

        } //End If statement for ages plus group
    } //End loop over ages
    for (f=1;f<=fisheries;f++)
    { //Start loop over fisheries
        // Calculate the number of tags caught in each region for each fishery
        TagsCaught(ty,s,ry,f)=sum(elem_prod(TagMvmnt[ty][s][ry],Baranov[ry][f]));
    } //End Loop over fisheries
} //End Loop over recapture years 1
    for (ry=(ty+ages);ry<=years;ry++)
{ //Begin loop over recapture years 2 to loop over the years that just have tags in the
plus group
    // Calculate the fish that move to each region after applying the tag shedding rate
    TagMvmnt[ty][s][ry][ages]=TagsAlive(ty,ry,ages,s)*(TagRetention((ry-ty+1),s))*Movement[s];
    if (ry<years)
        // Calculate the tags alive at the beginning of the next years just for the plus group
        TagsAlive(ty,(ry+1),ages,s)+=sum(elem_prod(TagMvmnt[ty][s][ry][ages],Survival[ry][ages
]));
    for (f=1;f<=fisheries;f++)
    { //Start loop over fisheries
        //Calculate the tags that are caught in each region for just the plus group
        TagsCaught(ty,s,ry,f)=sum(elem_prod(TagMvmnt[ty][s][ry][ages],Baranov[ry][f][ages]));
    } //End Loop over fisheries
} //End loop over recapture years 2
// Calculate the Tags that are returned and the total tags returned
TagReturns[ty][s]=elem_prod(TagsCaught[ty][s],ReportingRate);
TotalReturned(ty,s)=sum(elem_prod(TagsCaught[ty][s],ReportingRate));
} //End If statement for tags released not 0
    if (RewardTagsReleased(ty,s)>0)
{ //If there are actually tags released do calculations
    for (ry=ty;ry<(ty+ages);ry++)
    { //Loop over recapture years 1 starting from tag year and going only to the age where all
ages are in the plus group so don't need to do all of these calculations. Will run another loop
for just the plus group
        for (a=1;a<=ages;a++)
        { //Loop over Ages
            //Calculate the tags that move to each region after applying a tag shedding rate
            RewardTagMvmnt[ty][s][ry][a]=RewardTagsAlive(ty,ry,a,s)*(TagRetention((ry-ty+1),s))*
Movement[s];
            if (a<ages)
            { //Begin If loop for ages
                //Calculate the fish that are alive at the beginning of the next year
                RewardTagsAlive(ty,(ry+1),(a+1),s)=sum(elem_prod(RewardTagMvmnt[ty][s][ry][a],Survival[ry][
a]));
            }
            else
            { //Continue If statement
                //Calculate the fish that are alive at the beginning of the next year in
the plus group
                RewardTagsAlive(ty,(ry+1),ages,s)+=sum(elem_prod(RewardTagMvmnt[ty][s][ry
][ages],Survival[ry][ages]));
            } //End If statement for ages plus group
        } //End loop over ages
    } //Start loop over fisheries
    // Calculate the number of tags caught in each region for each fishery
    RewardTagsCaught(ty,s,ry,f)=sum(elem_prod(RewardTagMvmnt[ty][s][ry],Baranov[ry][f]));
} //End Loop over fisheries
    for (ry=(ty+ages);ry<=years;ry++)
{ //Begin loop over recapture years 2 to loop over the years that just have tags in the
plus group
    // Calculate the fish that move to each region after applying the tag shedding rate
    RewardTagMvmnt[ty][s][ry][ages]=RewardTagsAlive(ty,ry,ages,s)*(TagRetention((ry-ty+1),s))*
Movement[s];
    if (ry<years)
        // Calculate the tags alive at the beginning of the next years just for the plus group
        RewardTagsAlive(ty,(ry+1),ages,s)+=sum(elem_prod(RewardTagMvmnt[ty][s][ry][ages],
Survival[ry][ages]));
    for (f=1;f<=fisheries;f++)
    { //Start loop over fisheries
        //Calculate the tags that are caught in each region for just the plus group
        RewardTagsCaught(ty,s,ry,f)=sum(elem_prod(RewardTagMvmnt[ty][s][ry][ages],Baranov[ry][f][ages

```

```

    ));
  } //End Loop over fisheries
} //End loop over recapture years 2
// Calculate the Tags that are returned and the total tags returned
RewardTagReturns[ty][s]=RewardTagsCaught[ty][s];
RewardTotalReturned(ty,s)=sum(RewardTagsCaught[ty][s]);
} //End If statement for tags released >0
} //End Loop over stock of release
} //End Loop over tagging years

for (ty=(years-ages);ty<=years;ty++)
{
  //Loop over the last years to make sure that the array bounds are not exceeded
  // Initialize the Tags Alive as the number of tags released
  TagsAlive[ty][ty]=ReleaseAge[ty];
  RewardTagsAlive[ty][ty]=RewardReleaseAge[ty];
  for (s=1;s<=stocks;s++)
  {
    //Loop over stock of release
    if (TagsReleased(ty,s)>0)
  {
    //If there are actually tags released do calculations
    for (ry=ty;ry<=years;ry++)
    {
      //Loop over recapture years starting from tag year
      for (a=1;a<=ages;a++) //try getting rid of if statement
      {
        //Loop over Ages
        //Calculate the tags that move to each region after applying a tag shedding rate
        TagMvmnt[ty][s][ry][a]=TagsAlive(ty,ry,a,s)*(TagRetention((ry-ty+1),s))*Movement[s];
      if (ry<years)
      {
        //Begin If loop for recapture year
        if (a<ages)
        {
          //Begin If loop for ages
          //Calculate the fish that are alive at the beginning of the next year
          TagsAlive(ty,(ry+1),(a+1),s)=sum(elem_prod(TagMvmnt[ty][s][ry][a],
Survival[ry][a]));
        }
        else
        { //Continue If statement
          //Calculate the fish that are alive at the beginning of the next year in
the plus group
          TagsAlive(ty,(ry+1),ages,s)=sum(elem_prod(TagMvmnt[ty][s][ry][ages],
Survival[ry][ages]));
        } //End If statement for ages plus group
      } //End If statement for recapture year
    } //End loop over ages
    for (f=1;f<=fisheries;f++)
    { //Start loop over fisheries
      // Calculate the fish that are caught by each fishery
      TagsCaught(ty,s,ry,f)=sum(elem_prod(TagMvmnt[ty][s][ry],Baranov[ry][f]));
    } //End loop over fisheries
  } //End loop over recapture year
  // Calculate the Tags that are reported and the total tags returned
  TagReturns[ty][s]=elem_prod(TagsCaught[ty][s],ReportingRate);
  TotalReturned(ty,s)=sum(elem_prod(TagsCaught[ty][s],ReportingRate));
} //End If statement for tags released >0
if (RewardTagsReleased(ty,s)>0)
{
  //If there are actually tags released do calculations
  for (ry=ty;ry<=years;ry++)
  {
    //Loop over recapture years starting from tag year
    for (a=1;a<=ages;a++) //try getting rid of if statement
    {
      //Loop over Ages
      //Calculate the tags that move to each region after applying a tag shedding rate
      RewardTagMvmnt[ty][s][ry][a]=RewardTagsAlive(ty,ry,a,s)*(TagRetention((ry-ty+1),s))*
Movement[s];
      if (ry<years)
      {
        //Begin If loop for recapture year
        if (a<ages)
        {
          //Begin If loop for ages
          //Calculate the fish that are alive at the beginning of the next year
          RewardTagsAlive(ty,(ry+1),(a+1),s)=sum(elem_prod(RewardTagMvmnt[ty][s][ry]
[a],Survival[ry][a]));
        }
        else
        { //Continue If statement
          //Calculate the fish that are alive at the beginning of the next year in

```

```

the plus group
    RewardTagsAlive(ty, (ry+1), ages, s) += sum(elem_prod(RewardTagMvmnt[ty][s][ry]
[ages], Survival[ry][ages]));
    } //End If statement for ages plus group
} //End If statement for recapture year
    } //End loop over ages
    for (f=1; f<=fisheries; f++)
    { //Start loop over fisheries
        // Calculate the fish that are caught by each fishery
        RewardTagsCaught(ty, s, ry, f) = sum(elem_prod(RewardTagMvmnt[ty][s][ry], Baranov[ry][f]));
    } //End loop over fisheries
    } //End loop over recapture year
// Calculate the Tags that are reported and the total tags returned
    RewardTagReturns[ty][s] = RewardTagsCaught[ty][s];
    RewardTotalReturned(ty, s) = sum(RewardTagsCaught[ty][s]);
} //End If statement for reard tags released >0
    } //End loop over stock of release
} //End loop over tagging year
NotReturned = TagsReleased - TotalReturned;
RewardNotReturned = RewardTagsReleased - RewardTotalReturned;

// cout<<"Finished Calculate Tag Returns"<<endl;

FUNCTION CalculateObjectiveFunction
CatchNLL.initialize(); EffortNLL.initialize(); AgeCompNLL.initialize(); nll.initialize(); TagNLL.
initialize(); SurveyNLL.initialize(); SurveyAgeCompNLL.initialize(); InitAbunNLL.initialize();
RewardTagNLL.initialize();
double myeps=1.e-60;
double EPS=1.e-60;
if (current_phase() ==1) myeps=1.e-8;
//Calculate Sigma associated with the Effort data and Survey data
LogSigmaEffort=log(sqrt((1./EffortVarianceRatio)*square(mfexp(LogSigmaCatch))));
LogSigmaSurvey=log(sqrt((1./SurveyVarianceRatio)*square(mfexp(LogSigmaCatch))));
//Calculate the negative log likelihood for the total Catch
if (fisheries==4)
{
    for (f=1; f<=fisheries; f++)
    {
        CatchNLL += nllNormal(log(column(ObservedCatch, f)), log(column(TotalCatch, f)), exp(
LogSigmaCatch(f)));
    }
}
else
{
    for (f=1; f<=4; f++)
    {
        CatchNLL += nllNormal(log(column(ObservedCatch, f)), log(column(TotalCatch, f)), exp(
LogSigmaCatch(f)));
    }
    for (f=5; f<=fisheries; f++)
    {
        CatchNLL += nllNormal(log(column(ObservedCatch, f)), log(column(TotalCatch, f)), exp(
LogSigmaCatch(f-4)));
    }
}
//Calculate the negative log likelihood for the Survey
SurveyNLL = nllNormal(log(column(ObservedSurvey, 1)), log(column(TotalSurvey, 1)), exp(LogSigmaSurvey
(1)));
SurveyNLL += nllNormal(log(column(ObservedSurvey, 2)), log(column(TotalSurvey, 2)), exp(LogSigmaSurvey
(2)));
SurveyNLL += nllNormal(log(column(ObservedSurvey, 3)), log(column(TotalSurvey, 3)), exp(LogSigmaSurvey
(3)));
SurveyNLL += nllNormal(log(column(ObservedSurvey, 4)), log(column(TotalSurvey, 4)), exp(LogSigmaSurvey
(4)));
//Calculate the negative log likelihood associated with the age composition
AgeCompNLL = -sum(100.*elem_prod(ObservedAgeComp, log(AgeComp+myeps)));
//Calculate negative log likelihood associated with the survey age composition
SurveyAgeCompNLL = -sum(100.*elem_prod(ObservedSurveyAgeComp, log(SurveyAgeComp+myeps)));
//Calculate negative log likelihood associated with Effort Deviations
if (fisheries==4)
{
    for (f=1; f<=fisheries; f++)

```

```

    {
      EffortNLL+=nllNormal(column(LogEffortDevs , f) , zerovec2 , exp(LogSigmaEffort ( f) ));
    }
  }
else
{
  for ( f=1;f<=4;f++)
  {
    EffortNLL+=nllNormal(column(LogEffortDevs , f) , zerovec2 , exp(LogSigmaEffort ( f) ));
  }
  for ( f=5;f<=fisheries ; f++)
  {
    EffortNLL+=nllNormal(column(LogEffortDevs , f) , zerovec2 , exp(LogSigmaEffort ( f-4) ));
  }
}
// Calculate the negative log likelihood associated with the tag returns
for ( ty=1;ty<=years ; ty++)
{
  //Begin loop over tag years
  for ( s=1;s<=stocks ; s++)
  {
    //Begin loop over stocks
    if ( TagsReleased ( ty , s) >0)
    {
      //Begin if Tags release
      TagNLL-=sum( elem_prod ( log (( TagReturns [ ty ] [ s] +myeps) / ( TagsReleased ( ty , s) ) ) , TagsReported [ ty ] [ s] ));
      TagNLL-=log (( NotReturned ( ty , s) +myeps) / TagsReleased ( ty , s) ) * NeverRecovered ( ty , s );
    }
    //End if Tags released
  }
  //End loop over stocks
}
//End loop over tag years
//TagNLL-=sum( elem_prod ( log ( elem_div ( NotReturned +myeps , TagsReleased +myeps ) ) , NeverRecovered ) );

// Calculate the negative log likelihood associated with the reward tag returns
for ( ty=1;ty<=years ; ty++)
{
  //Begin loop over tag years
  for ( s=1;s<=stocks ; s++)
  {
    //Begin loop over stocks
    if ( RewardTagsReleased ( ty , s) >0)
    {
      //Begin if Tags release
      RewardTagNLL-=sum( elem_prod ( log (( RewardTagReturns [ ty ] [ s] +myeps) / ( RewardTagsReleased ( ty , s) ) ) , RewardTagsReported [ ty ] [ s] ));
      RewardTagNLL-=log (( RewardNotReturned ( ty , s) +myeps) / RewardTagsReleased ( ty , s) ) *
      RewardNeverRecovered ( ty , s );
    }
    //End if tags released
  }
  //End loop over stocks
}
//End loop over tag years
//RewardTagNLL-=sum( elem_prod ( log ( elem_div ( RewardNotReturned +myeps , RewardTagsReleased +myeps ) ) ,
//RewardNeverRecovered ) );
//Add in a recruitment penalty to help make the model converge
RecruitmentNLL=nllNormal(column(LogRecruitmentDevs , 1) , zerovec , exp(LogSigmaRec) );
RecruitmentNLL+=nllNormal(column(LogRecruitmentDevs , 2) , zerovec , exp(LogSigmaRec) );
RecruitmentNLL+=nllNormal(column(LogRecruitmentDevs , 3) , zerovec , exp(LogSigmaRec) );
RecruitmentNLL+=nllNormal(column(LogRecruitmentDevs , 4) , zerovec , exp(LogSigmaRec) );
//Calculated Process Error associated with Initial Abundance
InitAbunNLL=nllNormal(N01 , zerovec3 , exp(LogSigmaAbun) );
InitAbunNLL+=nllNormal(N02 , zerovec3 , exp(LogSigmaAbun) );
InitAbunNLL+=nllNormal(N03 , zerovec3 , exp(LogSigmaAbun) );
InitAbunNLL+=nllNormal(N04 , zerovec3 , exp(LogSigmaAbun) );

//Calculate Negative Log Likelihood
nll=CatchNLL+EffortNLL+AgeCompNLL+TagNLL+SurveyNLL+SurveyAgeCompNLL+RewardTagNLL+RecruitmentNLL+
InitAbunNLL;
//Add a likelihood term for the random walk of natural mortality if MEst==3
// if ( MEst==3)
// {
//   nll+=(LogSigmaM*size_count(LogMDevs) ) + (1./2.*square(mfexp(LogSigmaM) ) * norm2(LogMDevs) );
// }
//Add a likelihood term for the random walk of Reporting Rate if RREst==3
if ( RREst==3)
{
  nll+=(LogSigmaRR*size_count(LogRRDevs) ) + (1./2.*square(mfexp(LogSigmaRR) ) * norm2(LogRRDevs) );
}

```

RUNTIMESECTION

```

convergence_criteria 1.e-1,1.e-2,5.e-3
maximum_function_evaluations 5000,10000,15000,25000,50000

REPORT_SECTION
ofstream myreport ("release.txt");
myreport<<objective_function_value::pobjfun->gmax<<endl;
myreport<< "#Initial Abundance" <<endl;
myreport<< N[1] <<endl;
myreport<< "#True Initial Abundance" <<endl;
myreport<< TrueN0 <<endl;
myreport<< "#Initial Abundance Relative Error" <<endl;
for (a=2;a<=ages;a++)
    myreport<< elem_div((N[1][a]-TrueN0[a]),TrueN0[a])*100 <<endl;

myreport<< "#Mean Recruitment" <<endl;
myreport<< LogRecruits <<endl;
myreport<< "#True Mean Recruitment" <<endl;
myreport<< TrueMeanRecruits <<endl;
myreport<< "#Mean Recruitment Relative Error" <<endl;
myreport<< elem_div((LogRecruits-TrueMeanRecruits),TrueMeanRecruits)*100 <<endl;

myreport<< "#Recruitment Estimate" <<endl;
for (y=1;y<=years;y++)
    myreport<< N[y][1] <<endl;
myreport<< "#Recruitment True" <<endl;
myreport<< TrueRecruits <<endl;
myreport<< "#Recruits Relative Error" <<endl;
for (y=1;y<=years-2;y++)
    myreport<<elem_div((N[y][1]-TrueRecruits[y]),TrueRecruits[y])*100 <<endl;

myreport<< "#Catchability Coefficient" <<endl;
myreport<< Q <<endl;
myreport<< "#Catchability True" <<endl;
myreport<< TrueQ <<endl;
myreport<< "#Catchability Relative Error" <<endl;
for (f=1;f<=fisheries;f++)
{
    maxSel[f]=max(column(Selectivity,f));
}
for (y=1;y<=years;y++)
{
    myreport<< elem_div((elem_prod(Q[y],maxSel)-TrueQ),TrueQ)*100 <<endl;
}

myreport << "#Survey Catchability Coefficient" << endl;
myreport << mfexp(LogSurveyQ) << endl;
myreport << "#Survey Catchability True" << endl;
myreport << TrueSurveyQ << endl;
myreport << "#Survey Catchability Relative Error" << endl;
for (r=1;r<=regions;r++)
{
    maxSurveySel[r]=max(column(SurveySelectivity,r));
}
myreport << elem_div((elem_prod(mfexp(LogSurveyQ),maxSurveySel)-TrueSurveyQ),TrueSurveyQ)*100 <<
endl;

myreport<< "#Estimated Selectivity Matrix" <<endl;
myreport<< slctvty <<endl;
myreport<< "#Selectivity True" <<endl;
myreport<< TrueSel <<endl;
myreport<< "#Maximum Selectivity" << endl;
myreport<< maxSel << endl;
myreport<< "#Selectivity Relative Error" <<endl;
myreport<< elem_div((slctvty-TrueSel),TrueSel)*100 <<endl;
myreport<< "#Adjusted Selectivity Relative Error" << endl;
for (a=1;a<=ages;a++)
{
    myreport << elem_div((elem_div(slctvty[a],maxSel)-TrueSel[a]),TrueSel[a])*100 <<endl;
}
myreport<< ((1/maxSel)-1)/1*100 <<endl;

myreport << "#Estimated Survey Selectivity Matrix" << endl;

```

```

myreport << SrvySlctvty << endl;
myreport << "#Survey Selectivity True" << endl;
myreport << TrueSurveySel << endl;
myreport << "#Maximum Survey Selectivity" << endl;
myreport << maxSurveySel << endl;
myreport << "#Survey Selectivity Relative Error" << endl;
myreport << elem_div((SrvySlctvty-TrueSurveySel),TrueSurveySel)*100 << endl;
myreport << "#Adjusted Survey Selectivity Relative Error" << endl;
for (a=1;a<=ages;a++)
{
  myreport << elem_div((elem_div(SrvySlctvty[a],maxSurveySel)-TrueSurveySel[a]),TrueSurveySel[a]
  ))*100 <<endl;
}
myreport << ((1/maxSurveySel)-1)/1*100 <<endl;

myreport<< "#Movement Matrix" <<endl;
myreport<< Movement <<endl;
myreport<< "#Movement True" <<endl;
myreport<< TrueMvmnt <<endl;
myreport<< "#Movement Relative Error" <<endl;
myreport<< elem_div((Movement-TrueMvmnt),TrueMvmnt)*100 <<endl;

myreport << "#Log Sigma Catch" <<endl;
myreport << LogSigmaCatch << endl;
myreport << "#SigmaCatch Relative Error assuming 0.1" << endl;
myreport << (exp(LogSigmaCatch)-TrueSigmaCatch)/TrueSigmaCatch*100 << endl;

LastYearN=0;
for (a=1;a<=ages;a++)
  LastYearN += N[years][a];
myreport<< "#Last Years' Abundance summed over ages" <<endl;
myreport<< LastYearN <<endl;
myreport<< "#Last Years' Abundance True" <<endl;
myreport<< TrueLastYearN <<endl;
myreport<< "#Last Years' Abundance Error" << endl;
myreport<< elem_div((LastYearN-TrueLastYearN),TrueLastYearN)*100 <<endl;

if (PhaseRR>0 || RRVaryPhase>0)
{
  myreport<< "#Area Reporting Rate" <<endl;
  myreport<< ReportingRate <<endl;
  myreport<< "#Reporting Rate True" <<endl;
  myreport<< TrueRR <<endl;
  myreport<< "#Reporting Rate Relative Error" <<endl;
  myreport<< elem_div((ReportingRate-TrueRR),TrueRR)*100 <<endl;
}

if (PhaseM>0 || MVaryPhase>0)
{
  myreport<< "#Natural Mortality" <<endl;
  myreport<< M <<endl;
  myreport<< "#Natural Mortality True" <<endl;
  myreport<< TrueM <<endl;
  myreport<< "#Natural Mortality Relative Error" <<endl;
  myreport<< elem_div((M-TrueM),TrueM)*100 <<endl;
}

myreport<< "#Abundance at Age" <<endl;
for (s=1;s<=stocks;s++)
{
  for (y=1;y<=years;y++)
  {
    for (a=1;a<=ages;a++)
    {
      myreport<<N(y,a,s)<<" ";
    }
    myreport<<endl;
  }
}
myreport<< "#True Abundance at Age"<<endl;
for (s=1;s<=stocks;s++)
{

```

```

for (y=1;y<=years;y++)
{
  for (a=1;a<=ages;a++)
  {
    myreport<<NTrue(y,a,s)<<" ";
  }
  myreport<<endl;
}
}
myreport<< "#Abundance at Age Relative Error" <<endl;
NAll=0; NAllTrue=0; NTotal =0; NTotalTrue=0;
for (s=1;s<=stocks;s++)
{
  for (y=1;y<=years;y++)
  {
    for (a=1;a<=ages;a++)
    {
      myreport<<((N(y,a,s)-NTrue(y,a,s))/NTrue(y,a,s))*100<<" ";
      //Calculate Total abundnace and True total abundnace
      NAll(y,s)+=N(y,a,s);
      NAllTrue(y,s)+=NTrue(y,a,s);
      NTotal(y)+=N(y,a,s);
      NTotalTrue(y)+=NTrue(y,a,s);
    }
  }
  myreport<<endl;
}
}

myreport<< "#Abundance summed over ages" <<endl;
myreport<< NAll<<endl;
myreport<< "#True Abundance summed over ages" <<endl;
myreport<< NAllTrue <<endl;
myreport<< "#Stock Abundance Error" << endl;
myreport<< elem_div((NAll-NAllTrue),NAllTrue)*100 <<endl;

myreport<< "#Abundance summed over ages and stocks" <<endl;
myreport<< NTotal<<endl;
myreport<< "#True Abundance summed over ages and stocks" <<endl;
myreport<< NTotalTrue <<endl;
myreport<< "#Total Abundance Error" << endl;
myreport<< elem_div((NTotal-NTotalTrue),NTotalTrue)*100 <<endl;

myreport<< "#True Apical Fishing Mortality" <<endl;
myreport<< TrueF<<endl;

myreport.close();

```