

# Supplementary Material: Simulation testing the robustness of a multi-region tag-integrated assessment model that exhibits natal homing and estimates natural mortality and reporting rate

Matthew T. Vincent, Travis O. Brenden and James R. Bence

This document contains additional information that was not included in the main manuscript. The Methods section presents tables of parameter value that are age (Table 1) or region (Table 2) specific. Additionally, symbols and equations used to calculate the likelihoods used by the ITCAAN model are also included in the Methods sections. 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

Table 1: 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	0.35	0.98	1	0.7	0.5	0.5
Fishery selectivity region 2	0.4	1	0.9	0.8	0.8	0.7
Fishery selectivity region 3	0.1	0.6	0.65	0.7	0.8	1
Fishery selectivity region 4	0.01	0.13	0.35	1	1	1
Allocation of tags	5%	10%	20%	20%	20%	25%

Preliminary analyses found that estimating the coefficients of a Ricker stock-recruitment function as part of the ITCAAN model resulted in inaccurate parameter estimates and model convergence problems. Attempts at estimating the log-standard deviation for the random walk deviations as part of the ITCAAN model also were not successful, so there was little choice but to assume a value. We chose to use a log-variance equal to 4 for our ITCAAN model because there is high interannual variability in recruitment for walleye in Lake Erie and we did not want to constrain estimates.

The likelihood component equations used in the ITCAAN model are presented in Table 4. Lognormal distributions were assumed for the log-likelihoods for region-specific total fishery harvests in number of fish caught (Equation 4.7) and survey indices (Equation 4.9) and lognormal penalties for the catchability (Equation 4.3) and recruitment random walk deviations (Equation 4.10) and initial abundance-at-age white-noise deviations (Equation 4.5). The log-standard deviation of the harvest data for each fishery was an estimated parameter. The log-standard deviations of the fishing effort and survey indices of abundance were calculated based on assumed ratios of their variances relative to the estimated variance of the harvest data (Equation 4.8). The assumed ratios were equal to the actual ratios in variances from the operating model. The log-standard deviations for the recruitment and initial abundance deviations were set equal to 4.0, which was intended to provide only a weak restriction on parameter estimates (Equation 4.8). Age-composition data from the harvest and survey were assumed to be multinomially distributed

Table 2: 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 $\alpha$	2.41807	1.48449	1	0.34915
Ricker stock recruit $\beta$	1.29135e-6	3.0618e-8	10e-6	2.80287e-8
Survey catchability	1.5e-5	5e-6	2e-7	8e-7
Fishery catchability	2e-6	8e-6	3e-5	6e-5
Reporting rate	0.5	0.5	0.5	0.5
Natural mortality	0.32	0.32	0.32	0.32
$\mu$	0.4382821	0.1941417	0.2317251	0.1941417
$\rho$	0.8441864	0.8441864	0.8441864	0.8441864
$\sigma$	0.06019978	0.06019978	0.06019978	0.06019978
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
$\rho_s$	$\sim \text{Trunc.Normal}(0.466, 0.260, -0.99, 0.99)$			
$\sigma_s$	$\sim \text{Trunc.Normal}(0.777, 0.313, 0, \infty)$			

with effective samples sizes equal to 100, although this was reduced to 50 for some scenarios examining the effects of data quality (Equation 4.2 and 4.4).

## Performance Metrics

The calculation of the relative error for some 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 3: Symbols and descriptions of variables used in ITCAAN likelihood equations.

Symbol	Description	Model	Symbol	Description	Model
$y$	Subscript for year	Both	$a$	Subscript for age	Both
$A$	Subscript for oldest modeled age	Both	$l$	Subscript for release cohort unique by year and region	Both
$p$	Subscript for spawning population	Both	$r$	Subscript for region	Both
$F$	Age-specific fishing mortality	Both	$f$	Apical fishing mortality	Both
$v$	Age-specific fishery selectivity	Both	$V$	Age-specific survey selectivity	Both
$\rho_r$	Autoregressive coefficient in autoregressive fishing mortality	Operating	$\mu$	Mean fishing mortality in autoregressive process	Operating
$\sigma_r$	Variance of fishing mortality in autoregressive process	Operating	$\sigma_C^2$	Variance of harvest data observation error	Operating
$\sigma_I^2$	Variance of index data observation error	Operating	$\sigma_E^2$	Variance of effort data observation error	Operating
$M$	Natural mortality	Both	$Z$	Total mortality	Both
$\epsilon$	Autoregressive recruitment variation	Operating	$\delta$	Random annual deviation in recruitment	Both
$\alpha$	Parameter of Ricker stock-recruit function	Operating	$\beta$	Parameter of Ricker stock-recruit function	Operating
$w$	Weight at age	Operating	$m$	Maturity at age	Operating
$N$	Population abundance	Both	$S$	Survival	Both
$T_{p,r}$	Movement rate from population p to region r	Both	$R$	Number of tagged fish released	Both
$n$	Number of tagged fish alive	Both	$d$	Number of tags recovered	Both
$\phi^F$	Expected proportion of tagged cohort harvested	Operating	$\phi^S$	Expected proportion of tagged cohort to survive	Operating
$\phi^M$	Expected proportion of tagged cohort to die from natural mortality	Operating	$\Phi$	Vector of expected proportions of the fates of tagged cohorts	Operating
$t$	Number of tags captured by fishery F, die naturally M, or survive S	Operating	$\omega_{p,r}$	Multinomial logit parameter for estimating movement from population p to region r	ITCAAN
$I$	Survey abundance index	Both	$q$	Catchability coefficient	Both
$C$	Harvest in numbers caught	Both	$E$	Observed fishing effort	Both
$\zeta$	Observation error in harvest	Operating	$\gamma$	Observation error in effort	Operating
$\Gamma$	Mean abundance for first model year	ITCAAN	$\Delta$	Age-specific abundance deviations in first year	ITCAAN
$\Lambda$	Recruitment in the first year	ITCAAN	$\lambda$	Annual recruitment deviation	ITCAAN
$K$	Catchability parameter in the first year	ITCAAN	$\kappa$	Annual catchability deviation	ITCAAN
$P$	Harvest age composition	ITCAAN	$\eta$	Survey age composition	ITCAAN
$\theta$	Tag recovery proportions	ITCAAN	$\Upsilon$	Angler reporting rate of tags	ITCAAN
$\psi_C$	Standard deviation for harvest data component	ITCAAN	$\psi_I$	Standard deviation for survey data component	ITCAAN
$\psi_E$	Standard deviation for fishery catchability random walk	ITCAAN	$\tau$	Observation error in survey index	Operating
$\psi_N$	Standard deviation for abundance deviations in first model year	ITCAAN	$\psi_R$	Standard deviation for recruitment deviations random walk	ITCAAN
$ESS_C$	Effective sample size for harvest age composition	ITCAAN	$ESS_S$	Effective sample size for survey age composition	ITCAAN
$\sigma_p^2$	Recruitment innovation variance	Operating	$\rho_r$	Autoregressive coefficient in autoregressive recruitment	Operating

Table 4: Likelihood Equations and penalties used in the ITCAAN model.

Equation Number	Likelihood Equation and Penalty
(4.1)	$-ln(L_{TP}) = -\sum_l \sum_p \left[ \sum_y \left( \sum_g \sum_a \left( d_{l,p,a} \right) ln \left( \hat{\theta}_{l,p,y,g} \right) \right) + \sum_a \left( d^{l,p,a} \right) ln \left( \hat{\theta}_{l,p,NR} \right) \right]$
(4.2)	$-ln(L_{CP}) = -ESS_C \sum_y \sum_g \sum_a P_{y,g,a} ln \left( \hat{P}_{y,g,a} \right)$
(4.3)	$-ln(L_{Effort}) = \sum_y \sum_g ln(\psi_E \sqrt{2\pi}) + 0.5 \left( \frac{(ln(\text{mean}\hat{q}_{y,g}) - ln(\hat{q}_{y,g}))}{\psi_E} \right)^2$
(4.4)	$-ln(L_{SP}) = -ESS_S \sum_y \sum_a \sum_r \eta_{y,a,r} ln(\hat{\eta}_{y,a,r})$
(4.5)	$-ln(L_{N0}) = \sum_a \sum_p ln(\psi_N \sqrt{2\pi}) + 0.5 \left( \frac{(-\hat{\Delta}_{p,a})}{\psi_N} \right)^2$
(4.6)	$-ln(L_q) = \sum_y \sum_r ln(\psi_E \sqrt{2\pi}) + 0.5 \left( \frac{(-\hat{\kappa}_{y,p})}{\psi_E} \right)^2$
(4.7)	$-ln(L_{Harvest}) = \sum_y \sum_g ln(\psi_C \sqrt{2\pi}) + 0.5 \left( \frac{(ln(C_{y,g}) - ln(\hat{C}_{y,g}))}{\psi_C} \right)^2$
(4.8)	$\psi_E = \psi_C \quad \psi_I = \sqrt{2 * \psi_C^2} \quad \psi_R = 4 \quad \psi_N = 4 \quad ESS_c = 100 \quad ESS_S = 100$
(4.9)	$-ln(L_{Survey}) = \sum_y \sum_r ln(\psi_I \sqrt{2\pi}) + 0.5 \left( \frac{(ln(I_{y,r}) - ln(\hat{I}_{y,r}))}{\psi_I} \right)^2$
(4.10)	$-ln(L_{Rec}) = \sum_y \sum_p ln(\psi_R \sqrt{2\pi}) + 0.5 \left( \frac{(-\hat{\lambda}_{y,p})}{\psi_R} \right)^2$

## 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. 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 under estimated. 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 and only small changes in the IQR occur. 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 1.5 times the true value in the ITCAAN model. Incorrectly specifying the reporting rate also caused biases in 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. These results lead us to the conclusion that estimating the natural mortality parameter within the ITCAAN model would be preferable to specifying a value.

## Additional Figures

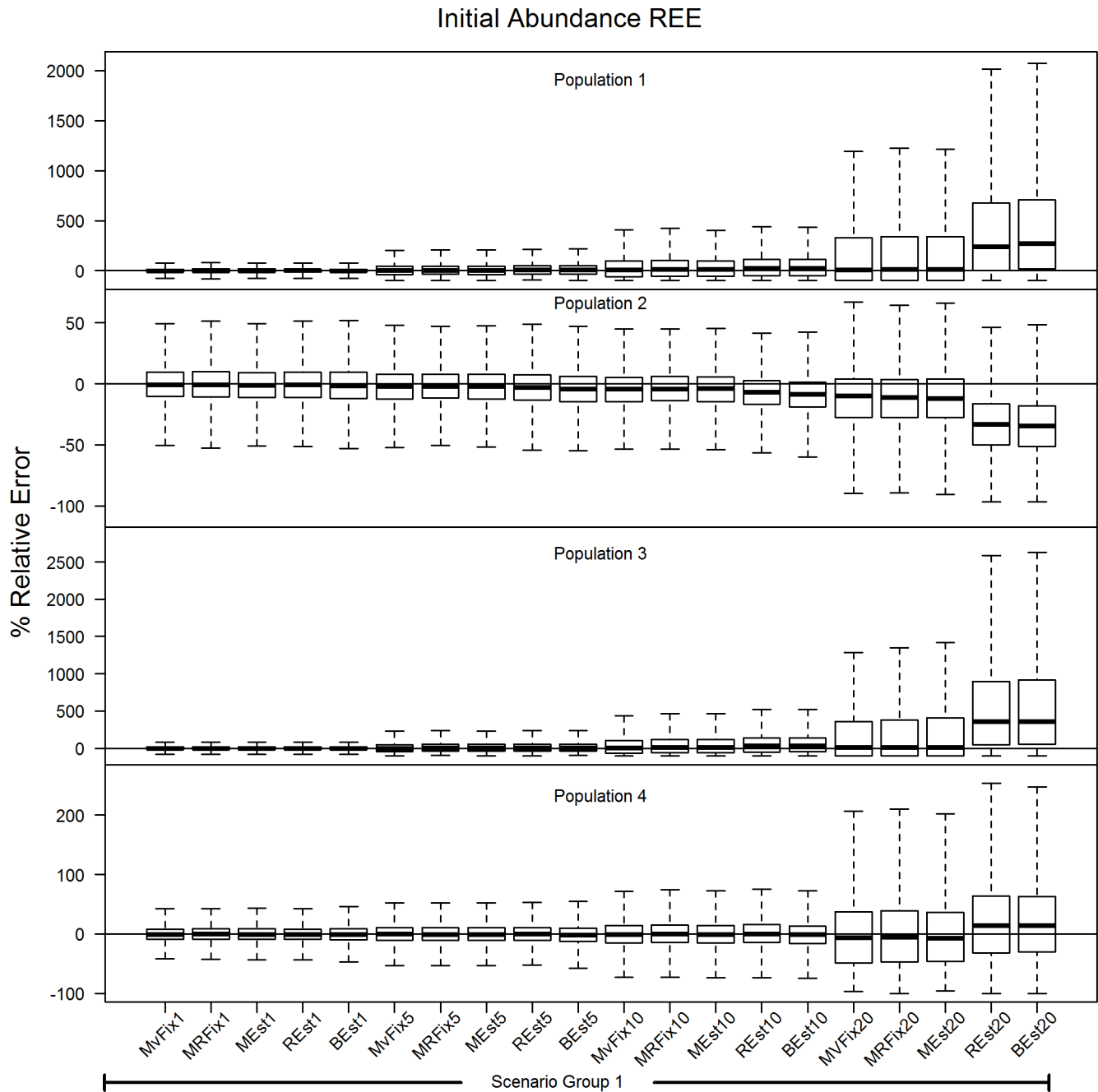


Figure 1: Relative error (%) of abundance at age in the first model year for each region of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

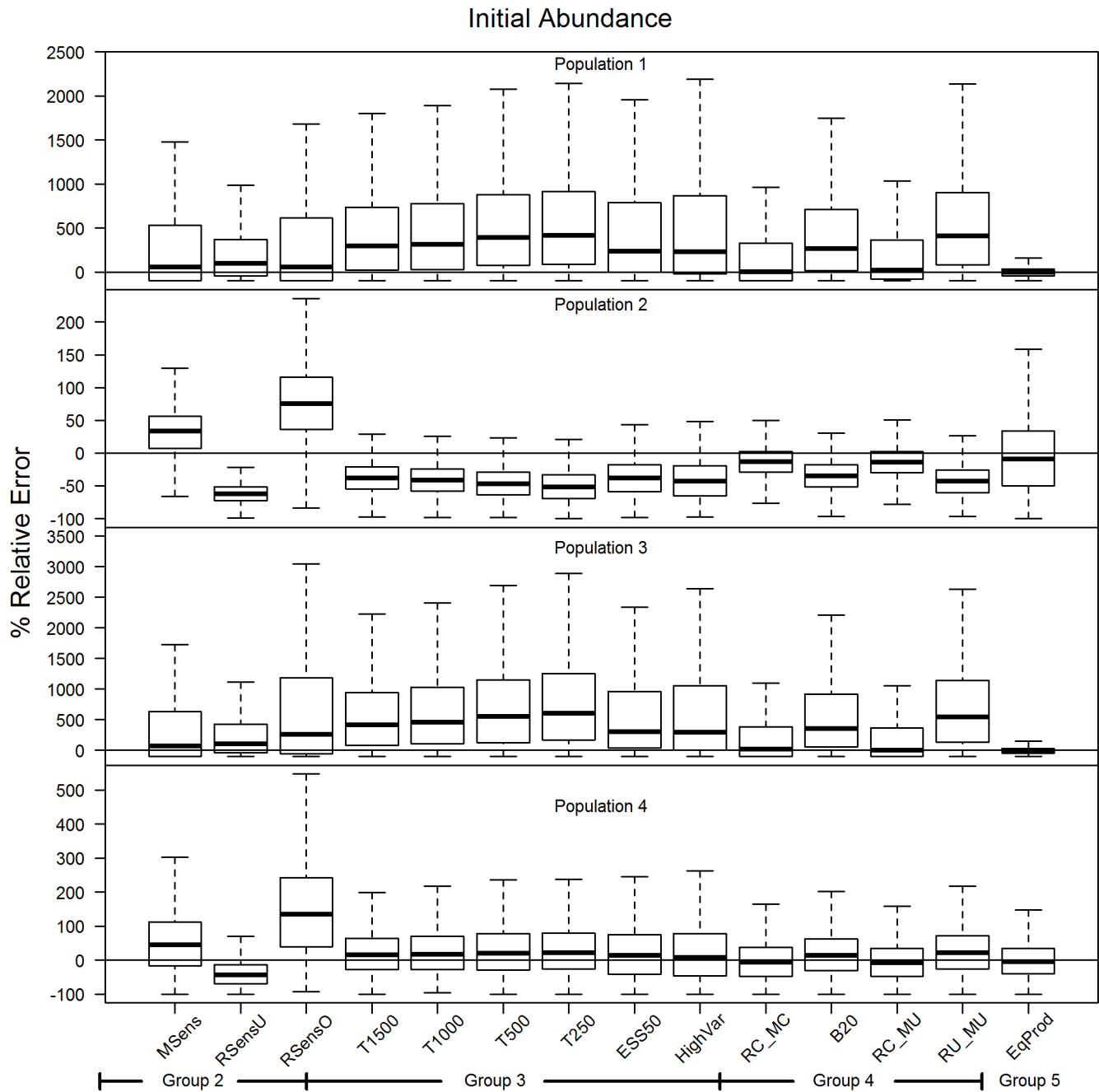


Figure 2: Relative error (%) of abundance-at-age in initial model year estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

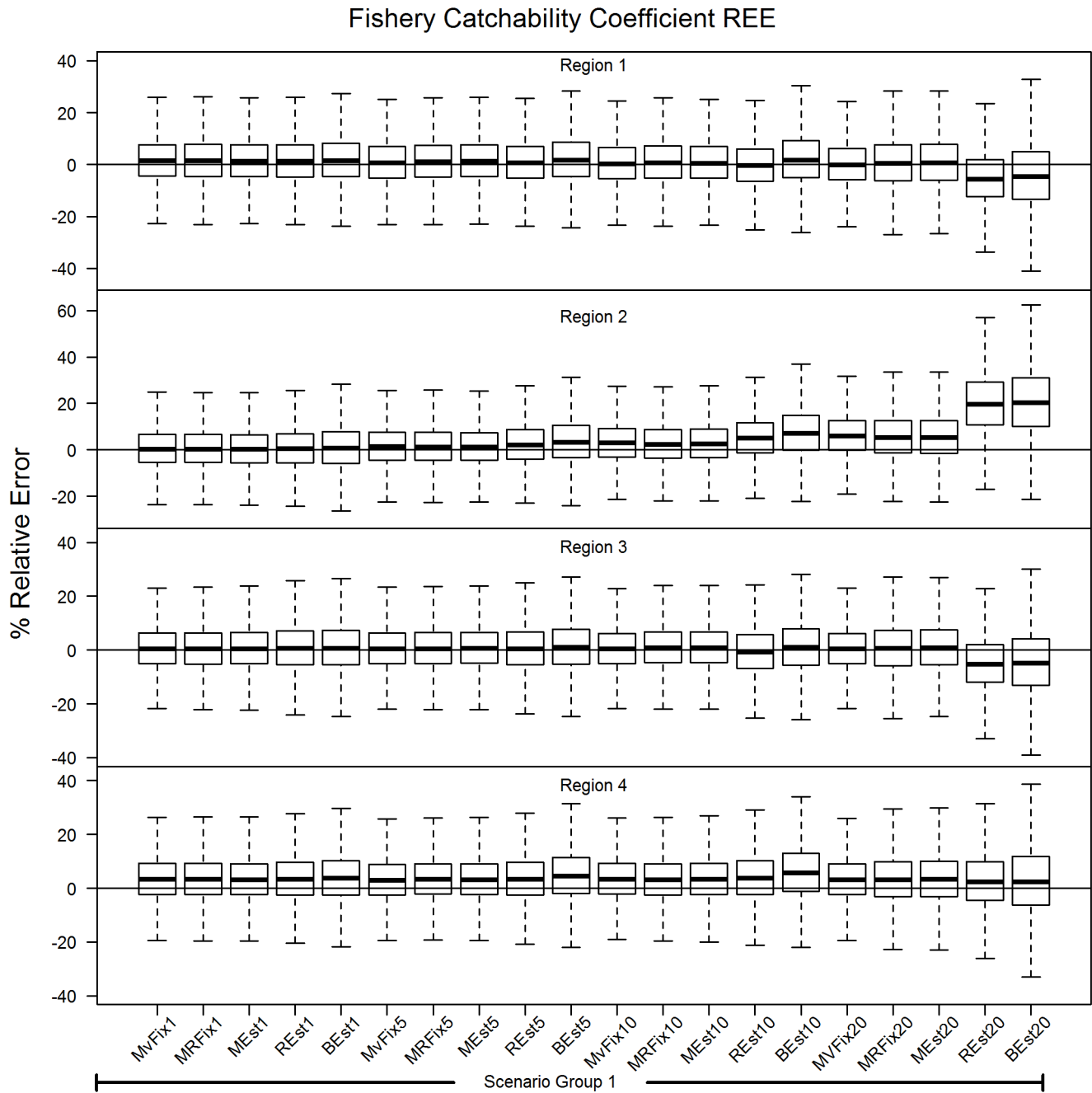


Figure 3: Relative error (%) of fishery catchability coefficients for each region of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

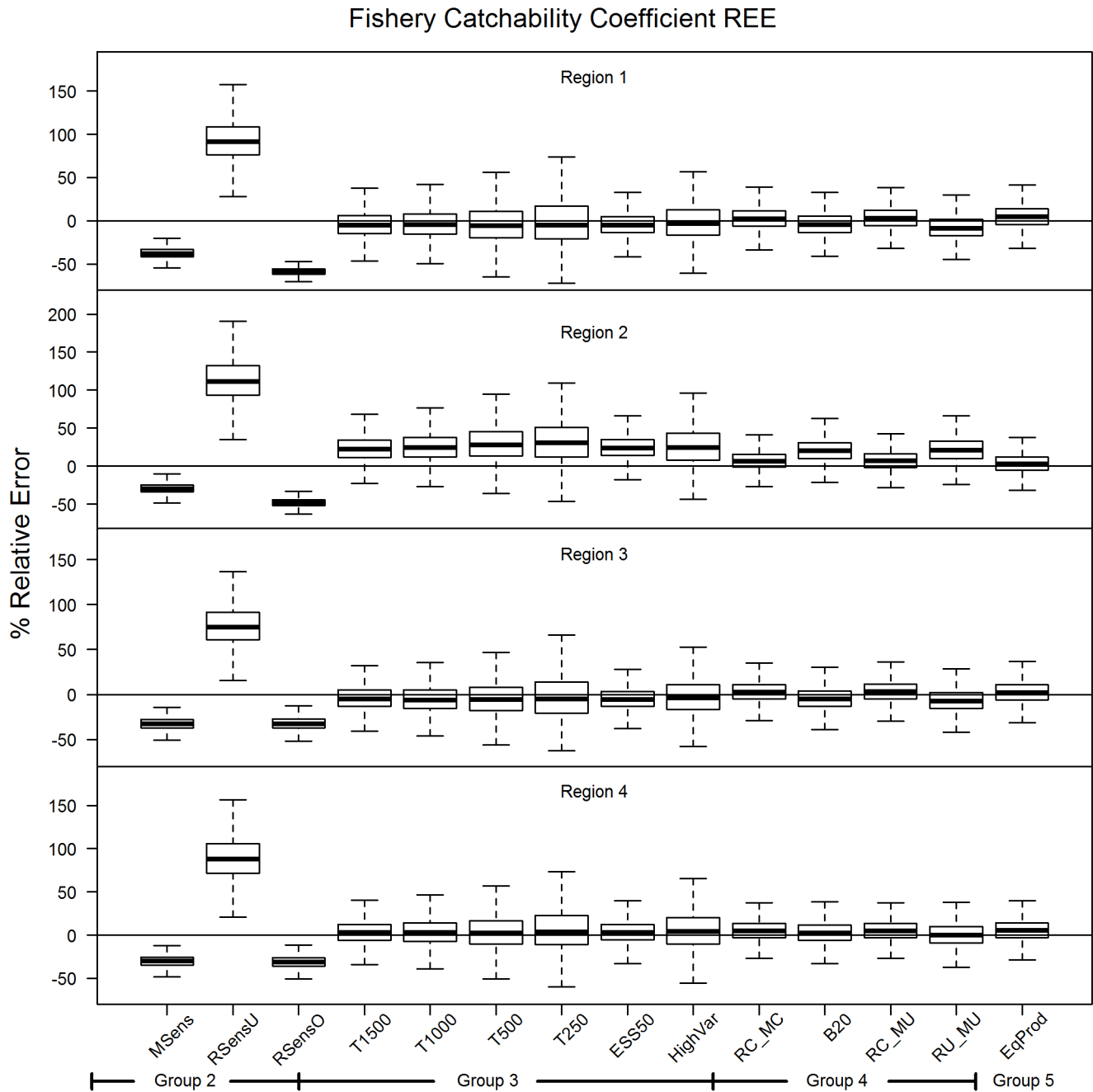


Figure 4: Relative error (%) of fishery catchability coefficients estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

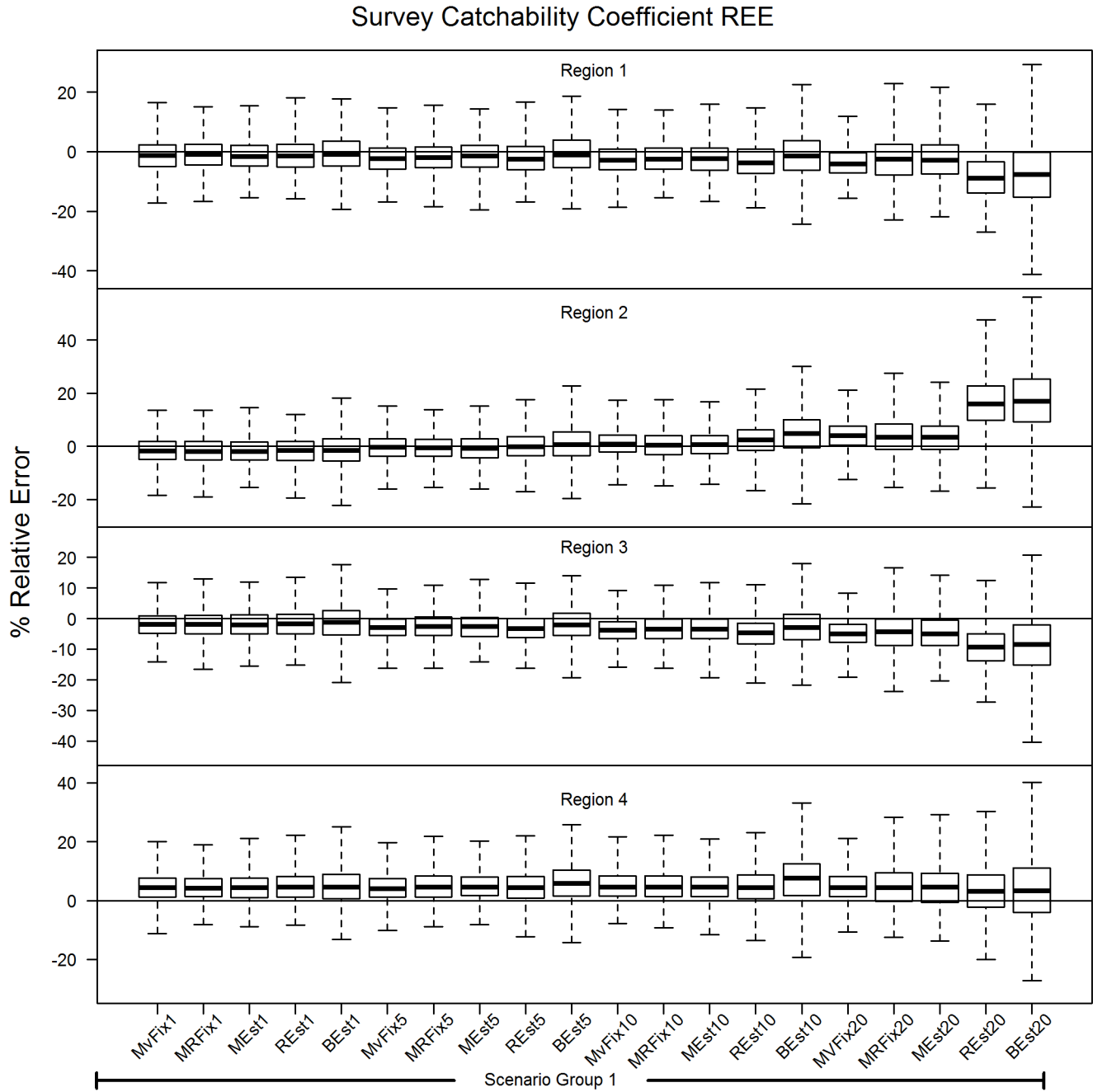


Figure 5: Relative error (%) of survey catchability coefficients for each region of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

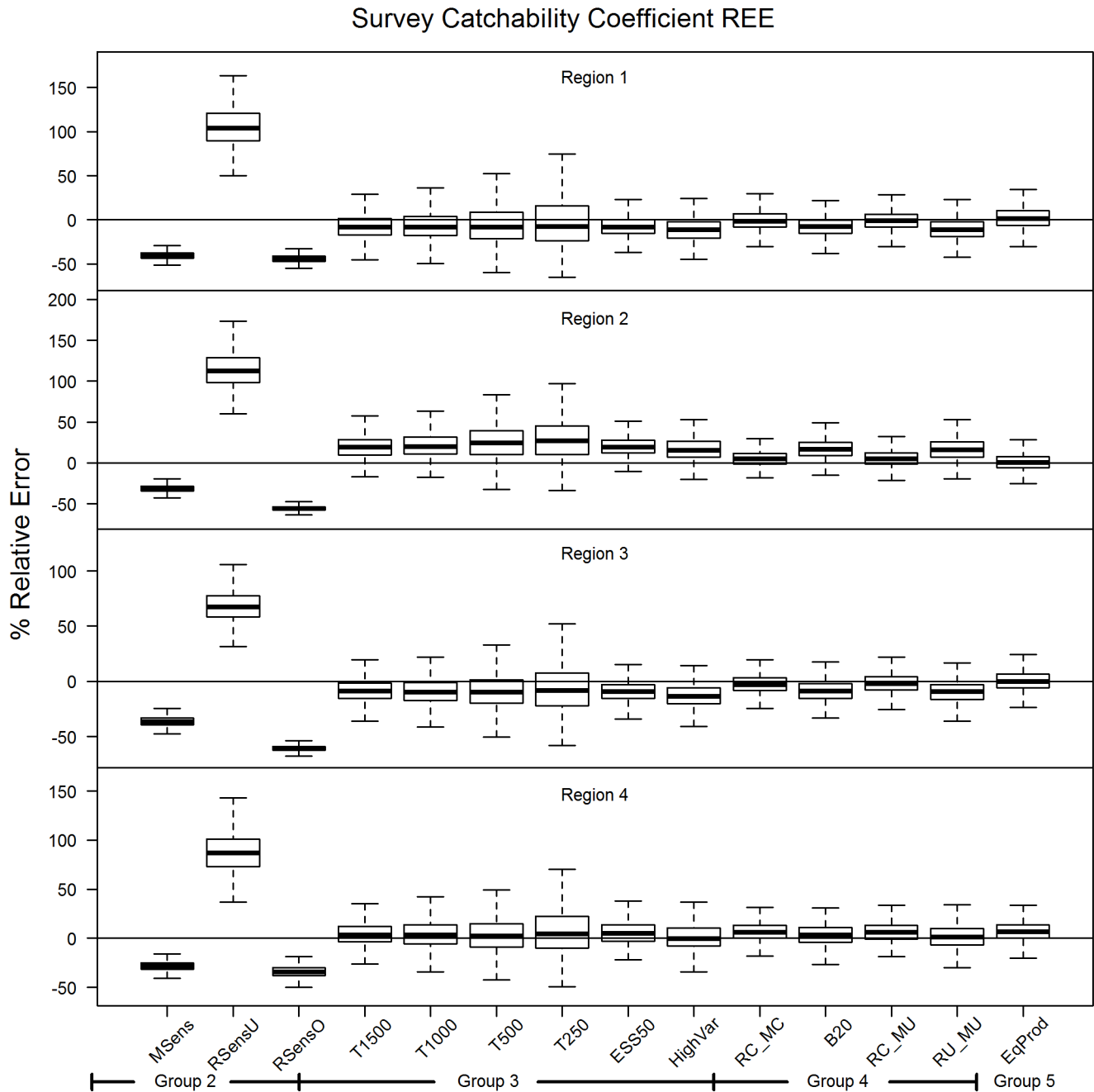


Figure 6: Relative error (%) of survey catchability coefficients estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

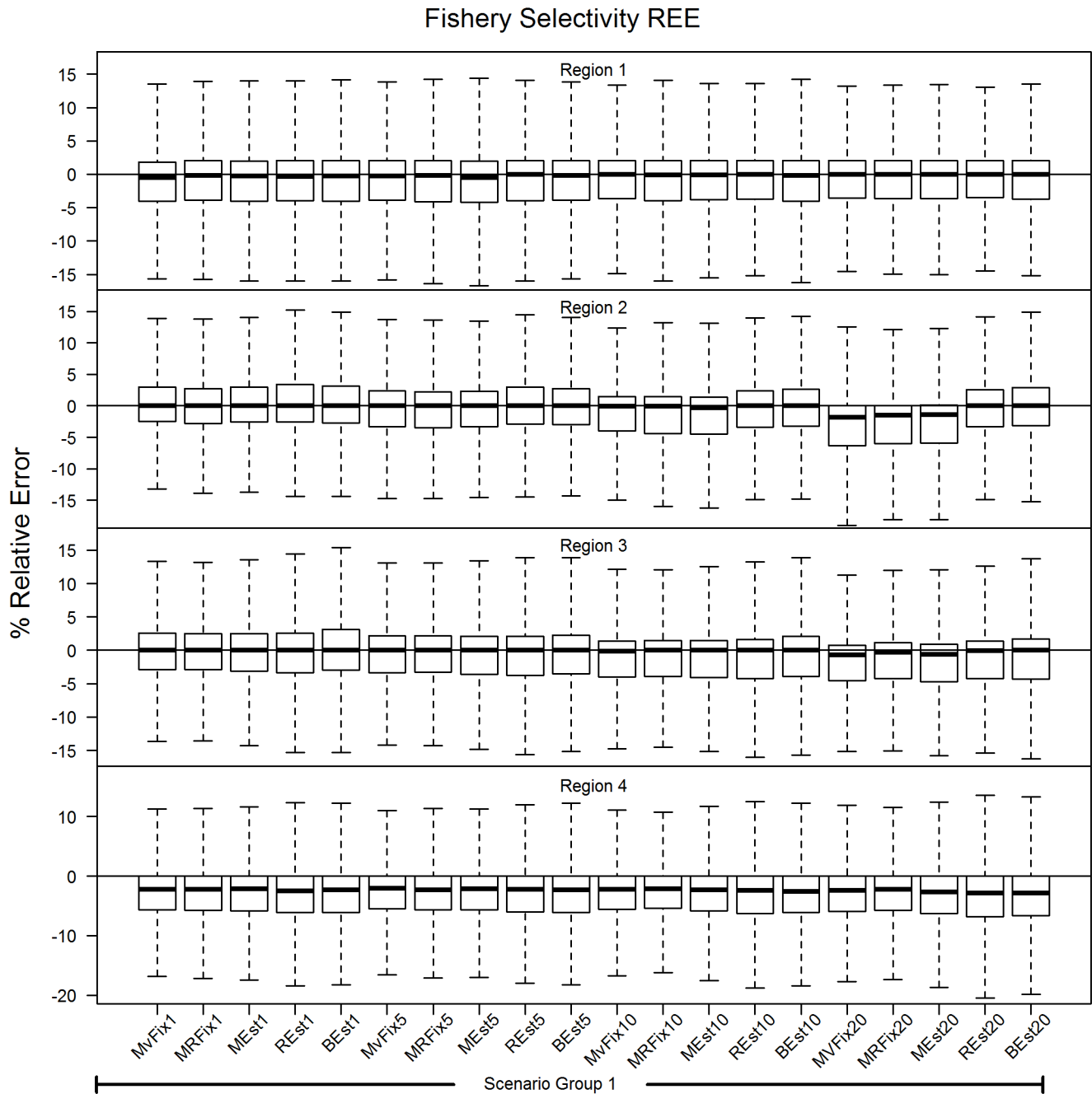


Figure 7: Relative error (%) of fishery selectivity for each region of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

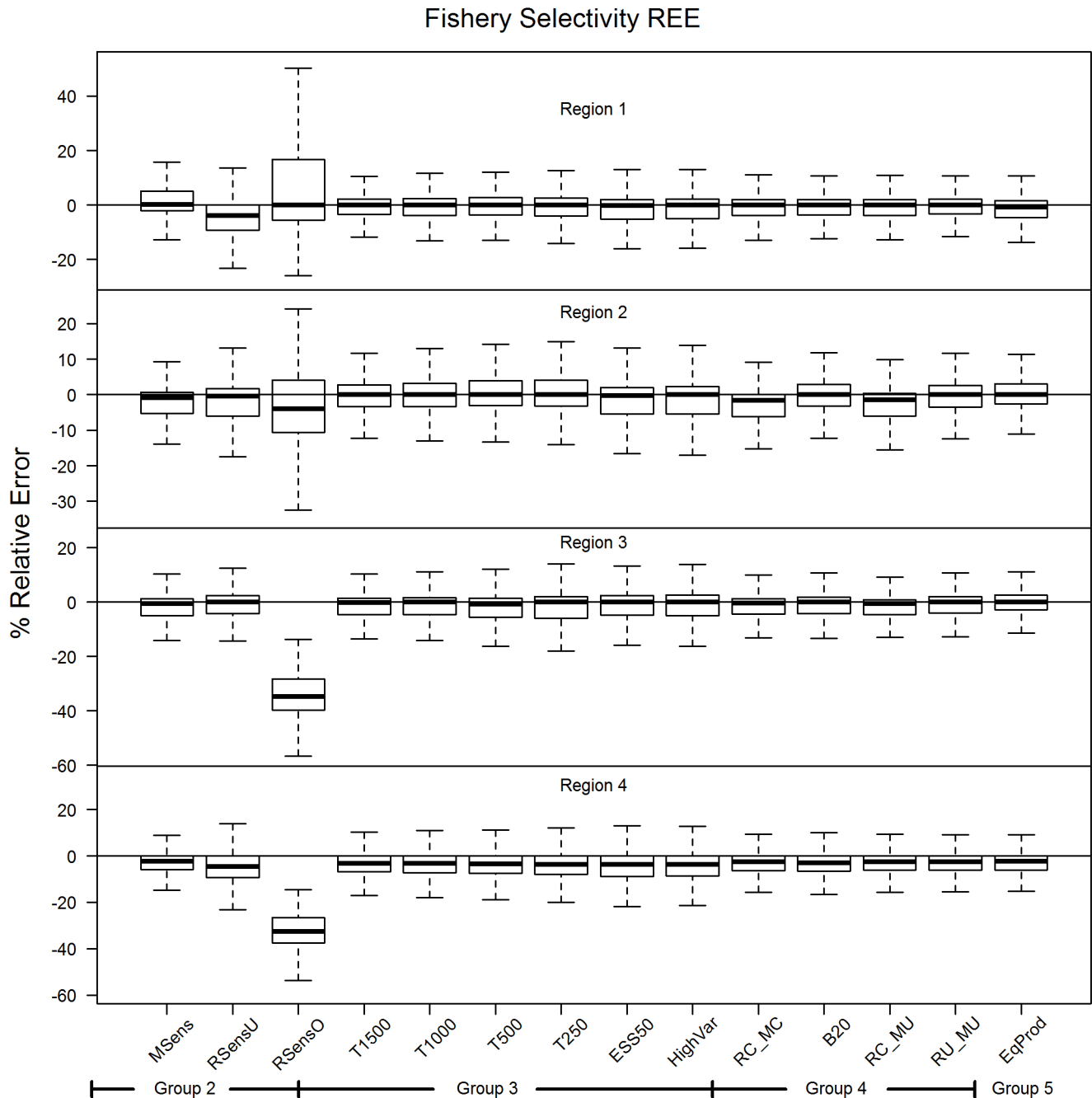


Figure 8: Relative error (%) of fishery selectivity estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

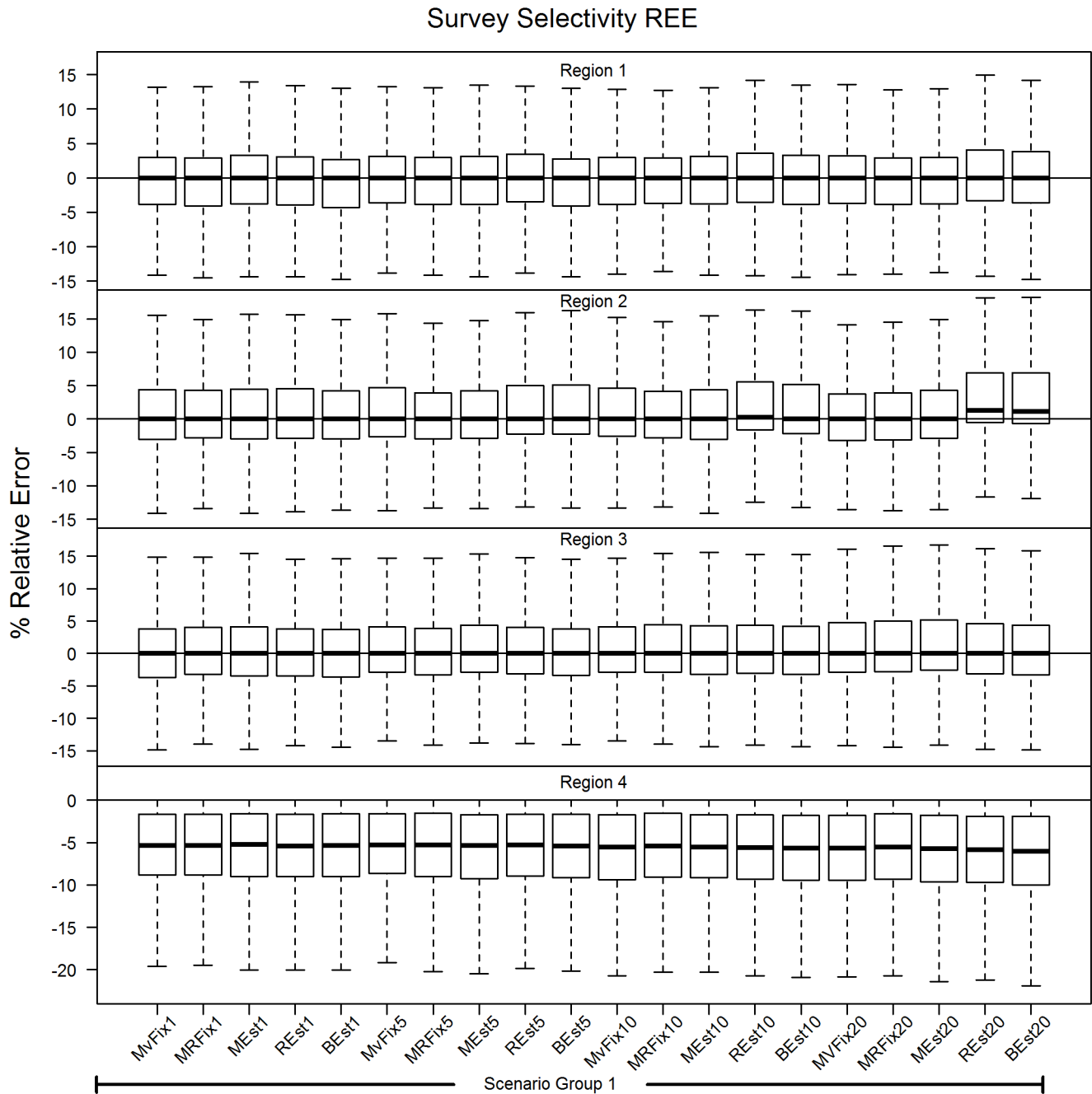


Figure 9: Relative error (%) of survey selectivity-at-age for each region of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

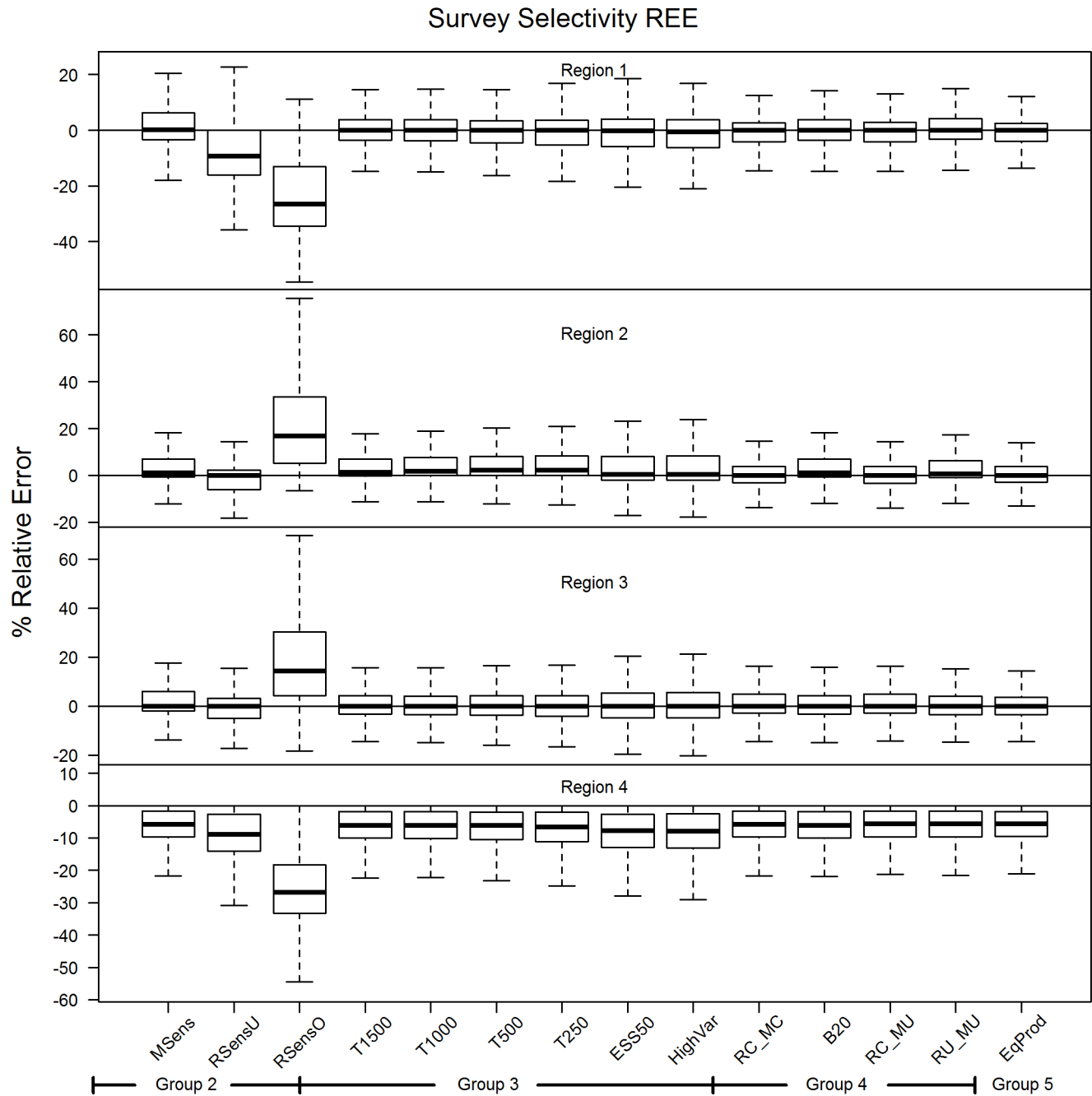


Figure 10: Relative error (%) of survey selectivity estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

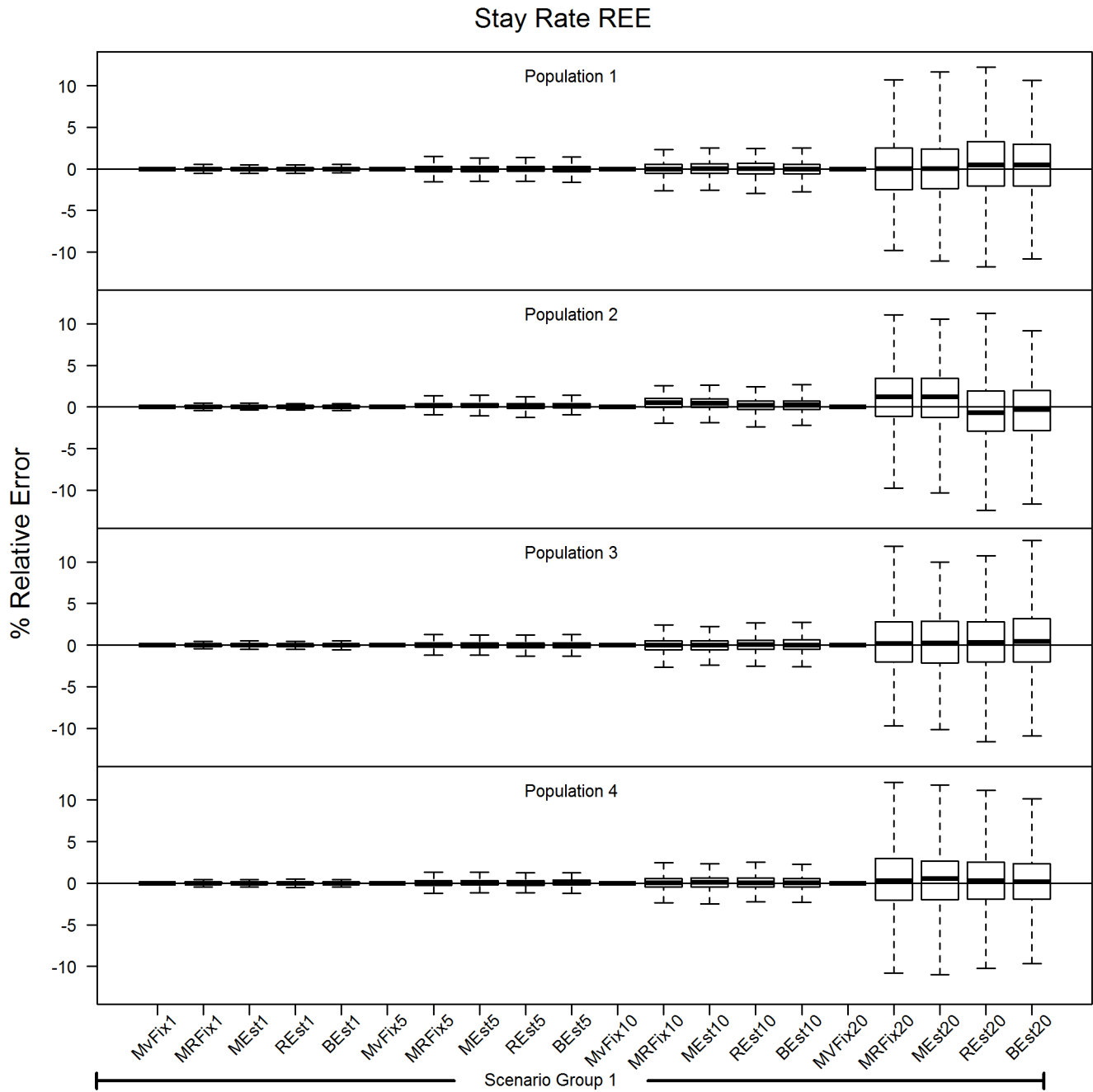


Figure 11: Relative error (%) of percent of population that remains in natal region (Stay Rate) for each population of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

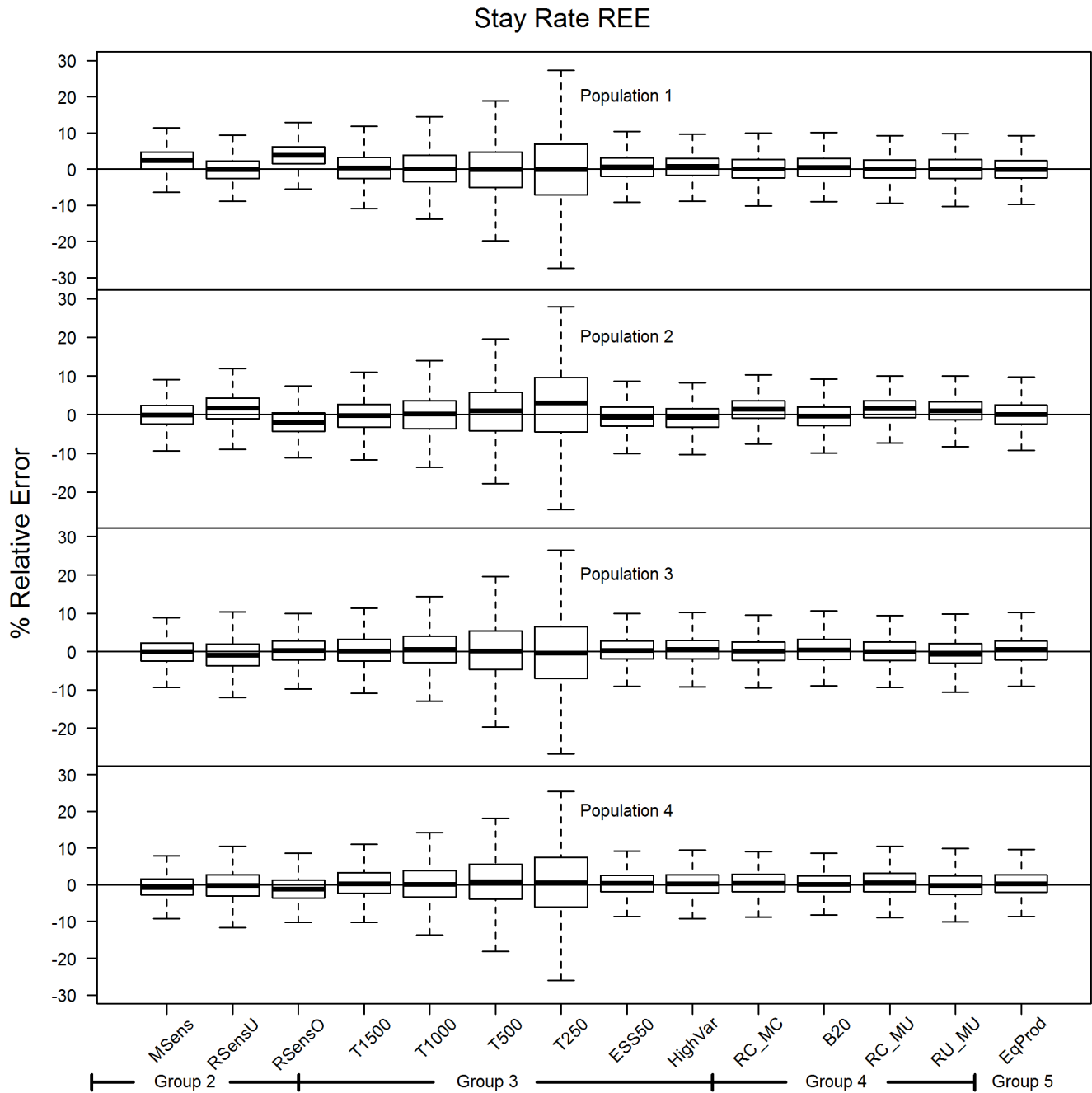


Figure 12: Relative error (%) of proportion of population that stays in natal region (Stay Rate) estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

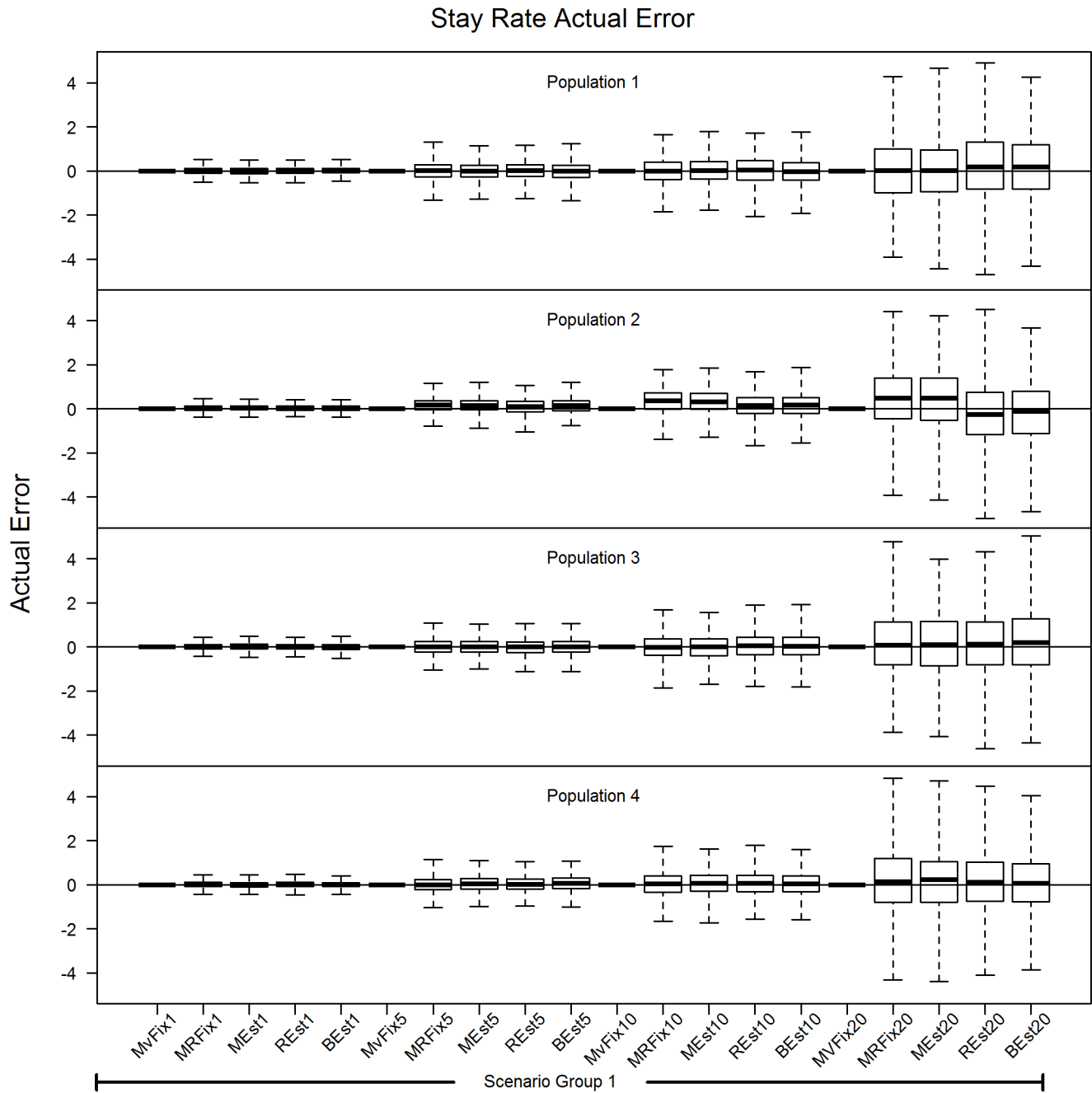


Figure 13: Actual error of estimates of percentage in population that remains in natal region (Stay Rate) for each population of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

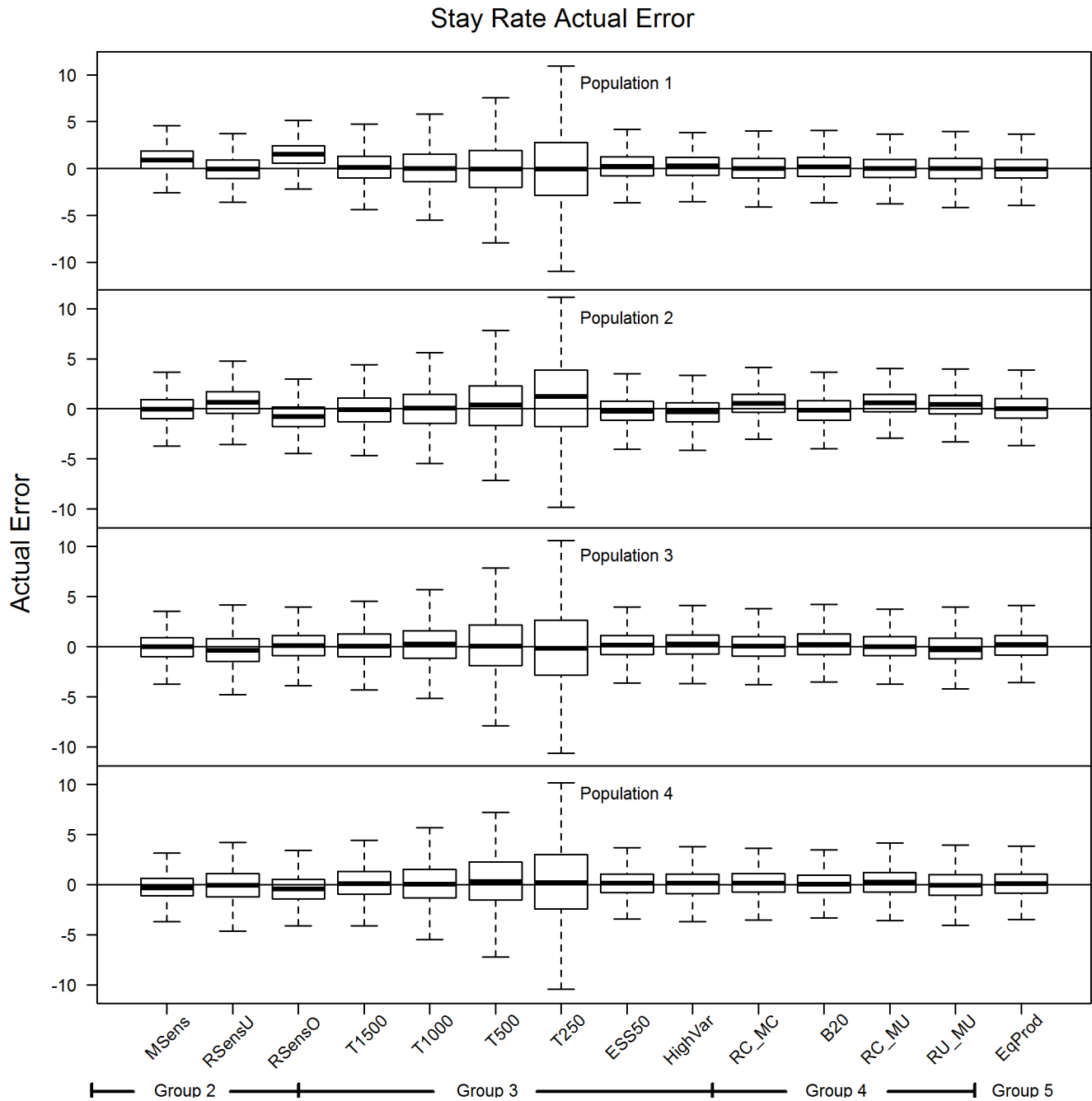


Figure 14: Actual error of proportion of population that stays in natal region (Stay Rate) estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

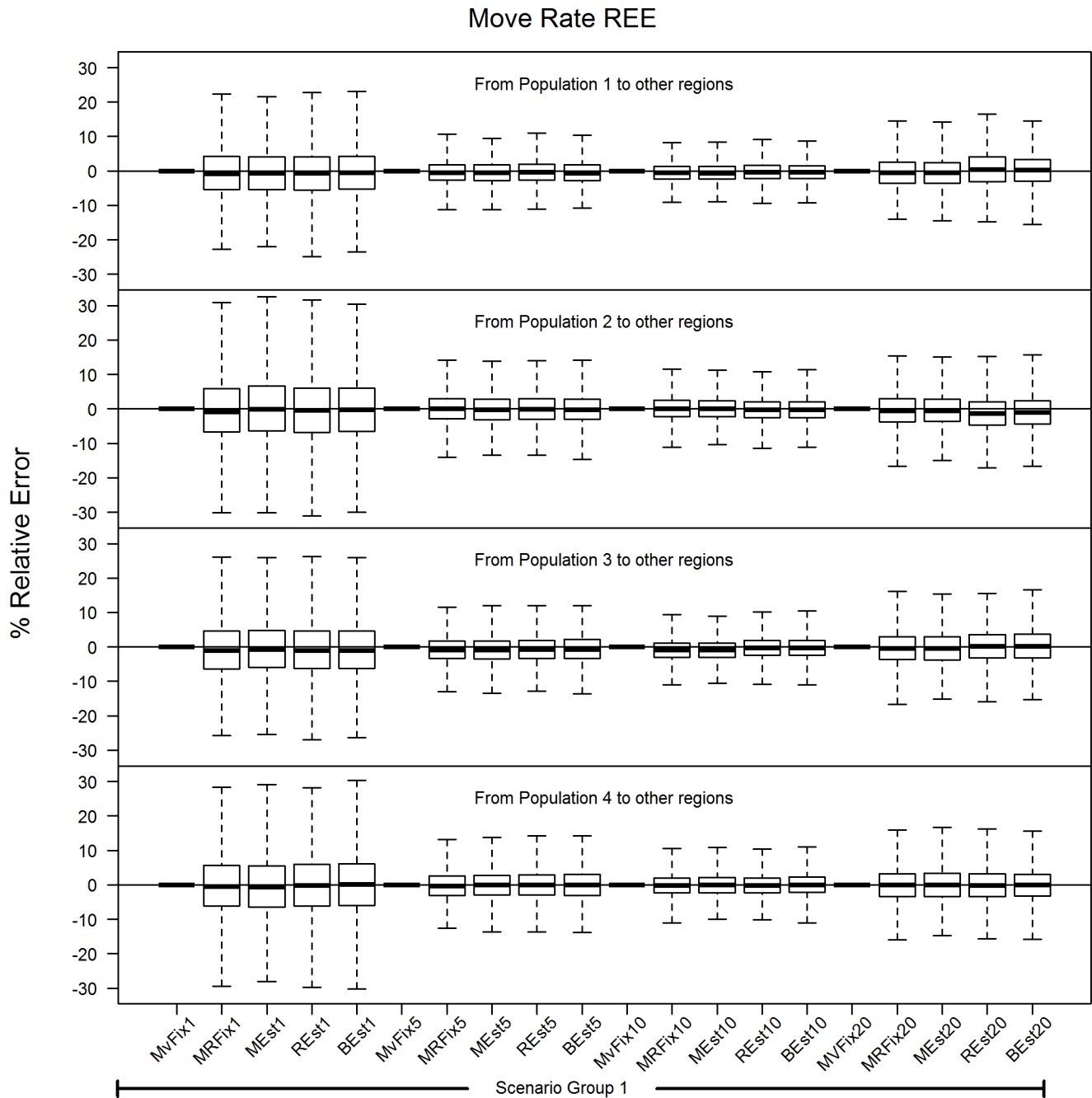


Figure 15: Relative error (%) of percent of population that move out of natal region (Move Rate) for each population of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

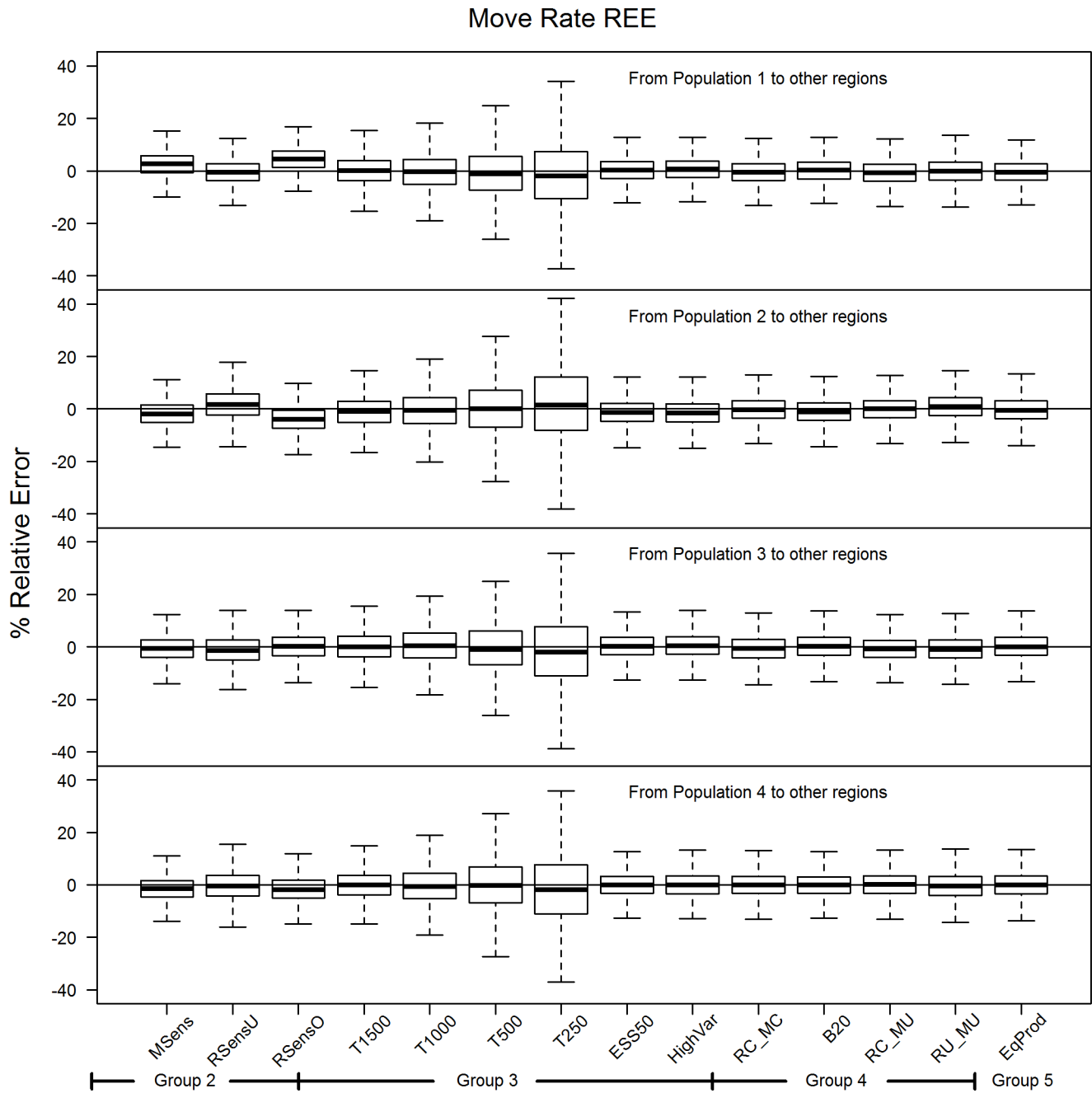


Figure 16: Relative error (%) of proportion of populations that moves to all other non-natal region (Move Rate) estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

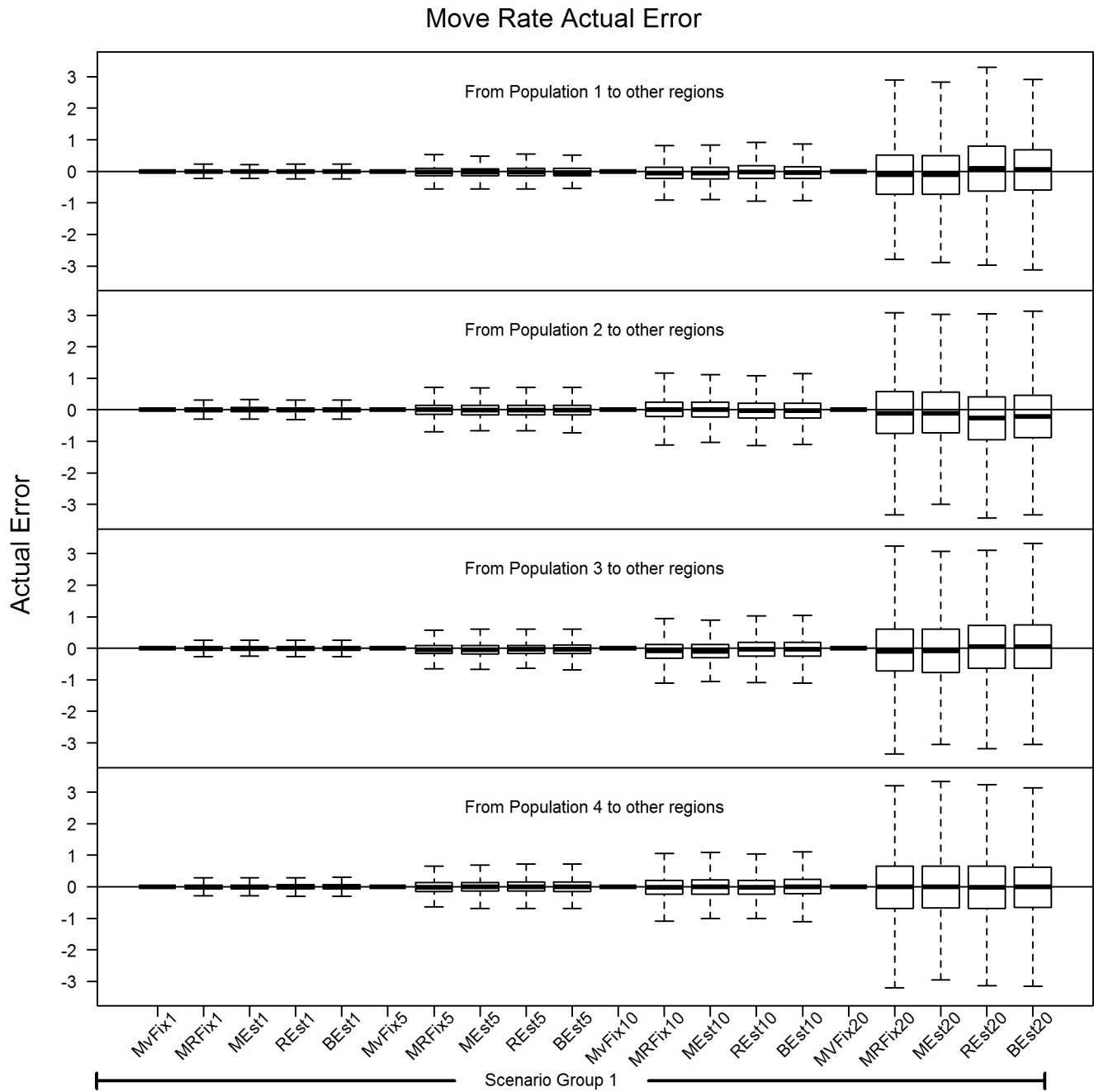


Figure 17: Actual Error of parameter estimates of proportion of population that move out of the natal region (Move Rate) for each population of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

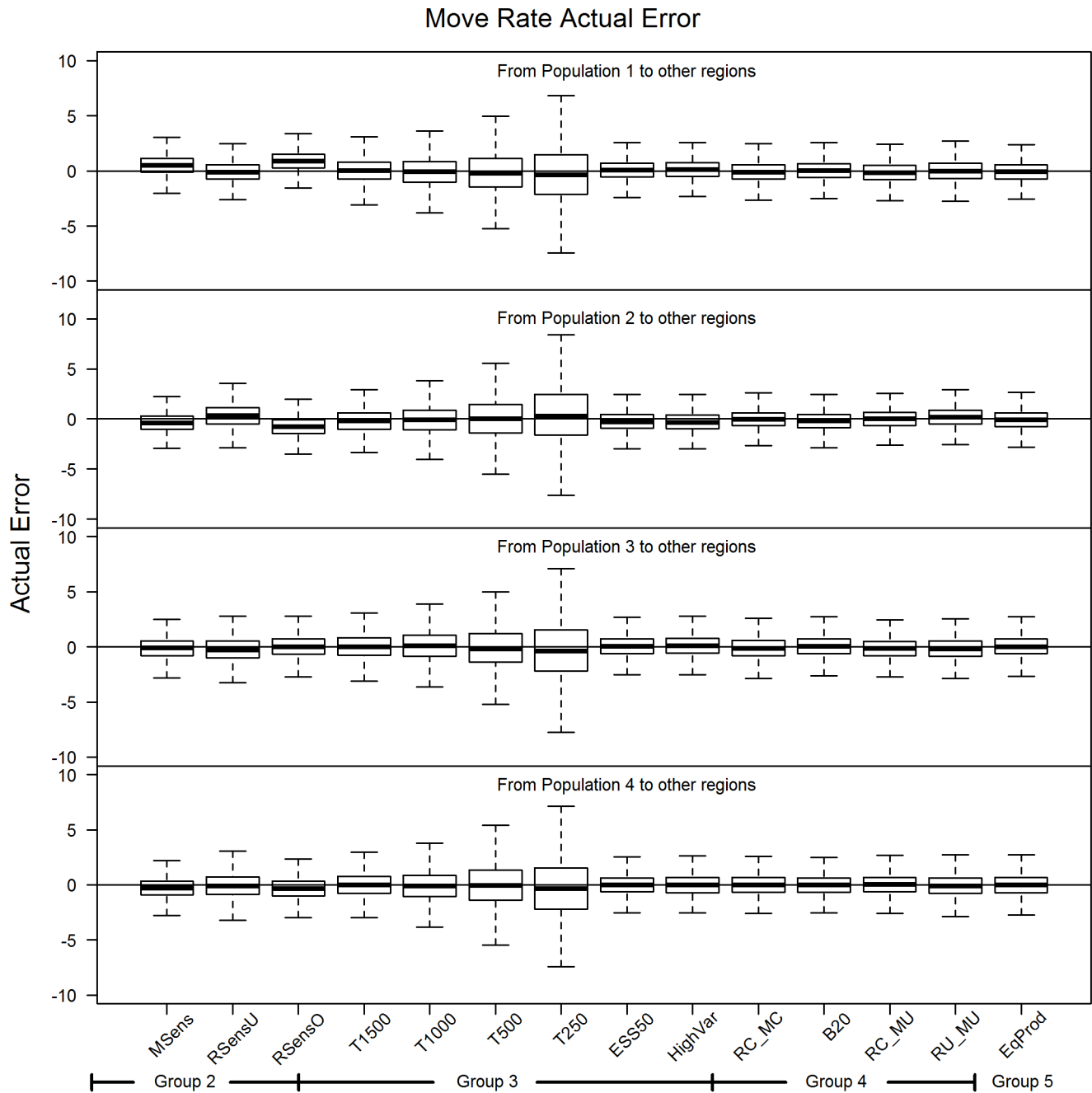


Figure 18: Actual error of proportion of populations that moves to all other non-natal region (Move Rate) estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

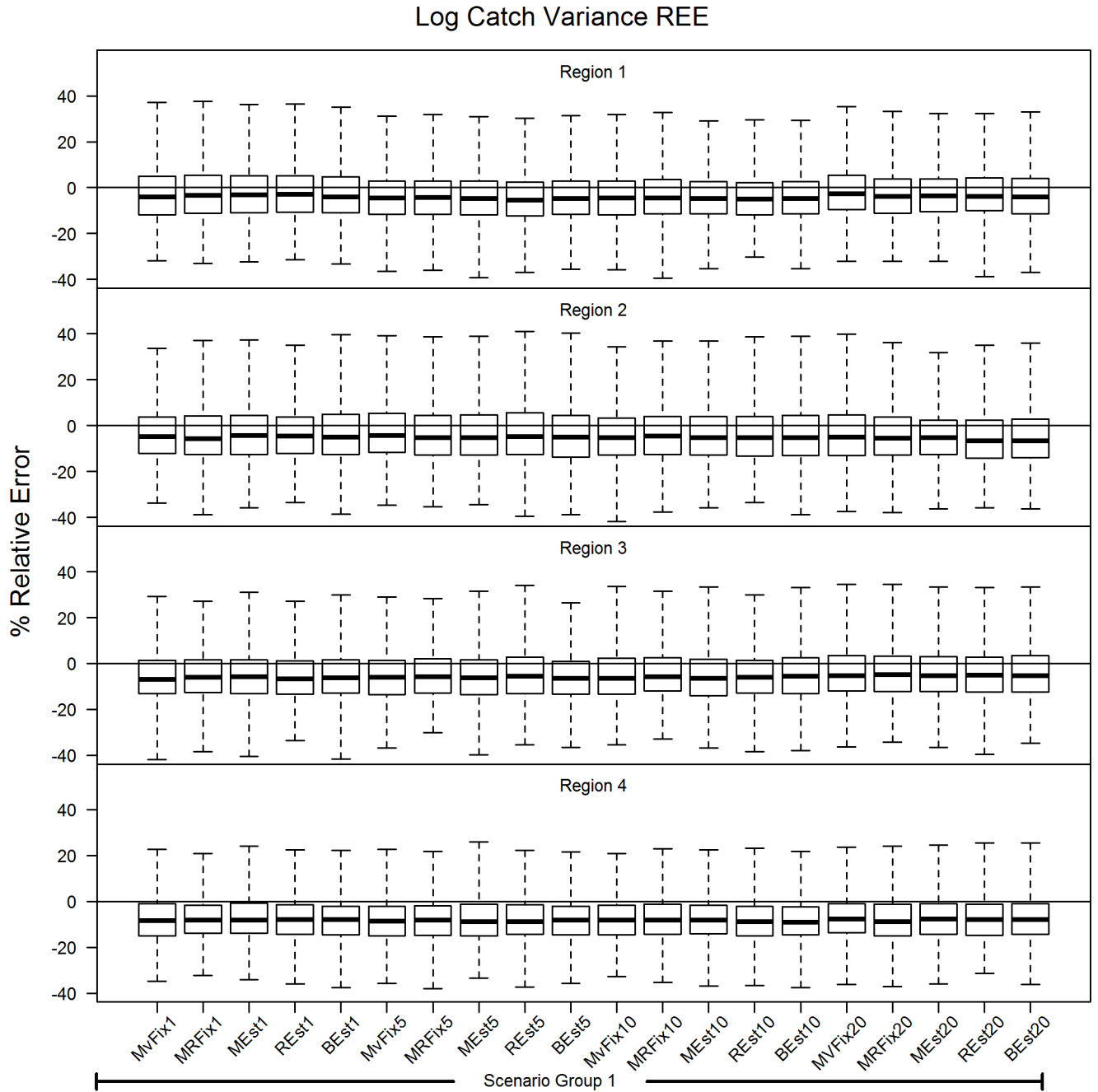


Figure 19: Relative error (%) of log-harvest variance of an ITCAAN model under different movement rates and parameter estimation assumptions (scenario group 1) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

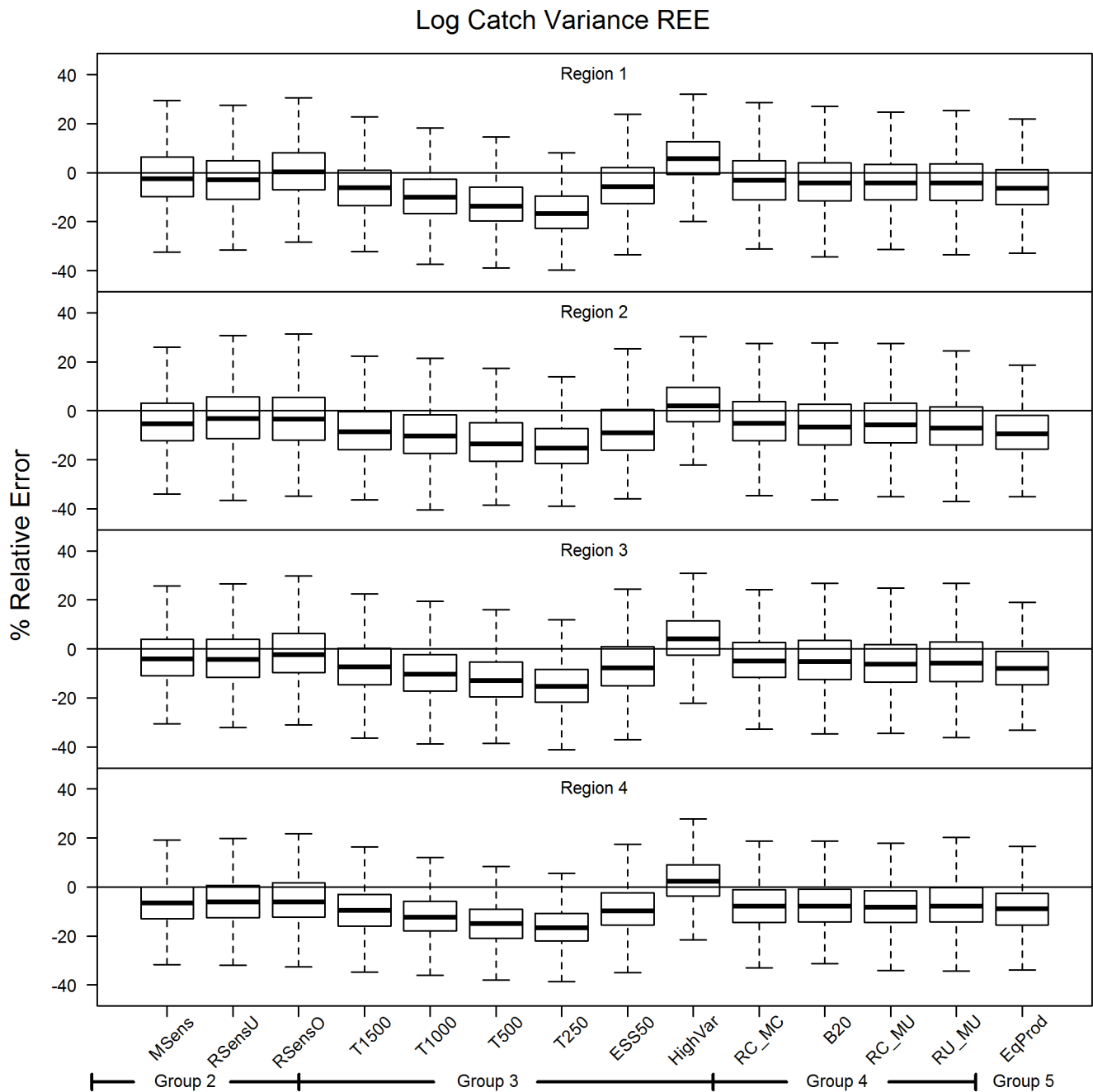


Figure 20: Relative error (%) of log variance of catch estimates for each region of an ITCAAN model under scenarios investigating misspecified natural mortality and reporting rates, data quality, spatial complexities, and equal productivities (scenario groups 2-5) for 1 000 simulation iterations. Table 1 lists the model abbreviations and corresponding model components. Whiskers on the boxplots extend to 1.5 times the inter-quartile range or the most extreme observed data point, whichever is less extreme. Data points outside the whisker range were not plotted. Note the difference in y-axis scale between populations.

## 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}
#Put in safety checks so that the correct simulation types are called
if (!(args[2]==1 | args[2]==2 | args[2]==3 | args[2]==4)) stop("RR Generation Type (args[2]) must
equal 1, 2, 3 or 4")
if (!(args[3]==1 | args[3]==2 | args[3]==3)) stop("RR Estimation Type (args[3]) must equal 1,2 or 3
")
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[10]==4 | args[10]==8)) stop("Number of fisheries (args[6]) must equal 4 or 8")
#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

#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,

##### 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]
```

```

##How often the Natural Mortality Parameter varies Should keep this at 5 since tpl is set for this
YrsMVary=5
#Natural mortality sensitivity scenario. If yes positive if no negative
MSensitivity=-1

##Time Varying Movement Phase of estimation
MvmntVaryPhase=-10
## Time Varying movement scenarios 0:Not time varying 1: Randomly varying Movement 2: linealy
  increasing movement out
MvmntTVType=0
#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: one tag loss 2: differnt tag loss in each region release
TagLossType=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 autocorrelation 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
TagLoss=numeric(length=stocks)

##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
  RR[1,]=rttruncnorm(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){
      RR[y,f]=c[f]+RRphi*RR[(y-1),f]+rtruncnorm(1,mean=0,sd=RRsigma,a=-(RRphi*RR[(y-1),f]+c),b

```

```

    =1-(c+RRphi*RR[(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)}

## 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$ar
  #Calculate the constant that needs to be added so that the mean is that of the regions
  cons=meanFs*(1-Fphi)
  #Set what the standard deviation of the process is based on wLE
  Fsigma=sqrt(z$var.pred)
  ## Randomly generate a starting F from a truncated normal distribution with lower bound 0
  effort[1,]=rtruncnorm(fisheries,meanFs,Fsigma,a=0,b=Inf)
  for (f in 1:fisheries){
    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 (fisheries==8){
  #Fishing mortality Trapnet Lake Huron
  meanFs[1]=mean(c
  (1.47358,0.0555759,0.171753,0.716091,0.0487051,0.0350967,0.0372921,0.691145,1.17786,0.0378429,0.0329804,0.
  )
  #Fishing mortality commercial gillnet Ontario western Lake Erie
  meanFs[2]=mean(c
  (0.0470186,0.0709162,0.0728645,0.0815978,0.0897574,0.100577,0.0711576,0.0469349,0.0634911,0.0556744,0.0631
  )
  #Fishing mortality commercial central Lake Erie Use commercial fishery for Western Basin
  meanFs[3]=mean(c
  (0.0470186,0.0709162,0.0728645,0.0815978,0.0897574,0.100577,0.0711576,0.0469349,0.0634911,0.0556744,0.0631

```

29

```

#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){N0R1= c(9000000,7000000,5000000,3000000,1000000,1500000)}
if (stocks>=2){N0R2= c(1000000,800000,600000,400000,200000,90000)}
if (stocks>=3){N0R3= c(500000,300000,100000,80000,50000,50000)}
if (stocks>=4){N0R4= c(500000,300000,100000,80000,60000,90000)}
if (stocks>=5){N0R5= 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
NMvmnt=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){
    #Calculate the 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()
    }
  } #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 separeate
      #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,f,r]/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) error("I never set the catchability for 8 fisheries")

#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=TagsAlive[ty,ry,a,s]*(1-TagLoss[s]),
        prob=Movement[s,])
        #check to make sure tags aren't created or destroyed during movement
        if(round(TagsAlive[ty,ry,a,s]*(1-TagLoss[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,a,r]/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,f,r]/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]*(1-TagLoss[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])
    }
}

#####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"
,M[1]), 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")

if(MVaryPhase>0){
  cat("#This is the True Time Varying Natural Mortality \n \n",append=TRUE, file="SimulatedData.
  dat")
  write.table(t(M),append=TRUE, file="SimulatedData.dat",sep=" ",row.names = FALSE, col.names =
  FALSE)
}

#Prints out Tag Loss
cat(c("#This is the Tag Loss as a decimal yearly percentage lost","\n",TagLoss),file="SimulatedData
  .dat",append = TRUE,sep=" ")
#write.table(TagLoss,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
#ObservedAgeComp1=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=fisheries)
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){
  cat(c("\n", "#This is the True Mean Reporting Rate", "\n", "\n"), file="SimulatedData.dat", append=
    TRUE, sep=" ")
  write.table(ReportingRate[1,], file="SimulatedData.dat", append=TRUE, row.names = FALSE, col.names
    = FALSE, sep=" ")
}

#Print out the True Time Varying Reporting rate only if it is estimated
if(RRVaryPhase>0){
  cat(c("\n", "#This is the True Annual 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("\n", "#This is the second test number", 1234567890, "\n"), file="SimulatedData.dat", append = TRUE,
  sep="\n")

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

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

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

#Print out test number 4
cat(c("\n", "#This is the fourth 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 5
cat(c("\n", "#This is the fifth 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=.1

```

```

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("# SrvySlctvty \n", file="release.pin", append=TRUE)
write.table(StartSrvySlctvty, file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col.names =

```

```

FALSE)
#cat("# SrvySlctvty \n", file="caa.pin", append=TRUE)
#write.table(StartSrvySlctvty, 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), rr, abs(rr*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="recapture.pin", append=TRUE)
  write.table(matrix(0, ncol=fisheries, nrow=1), file="recapture.pin", sep=" ", append=TRUE, row.names
= FALSE, col.names = FALSE)
  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="recapture.pin", append=TRUE)
  write.table(matrix(0, ncol=fisheries, nrow=(years/5)), file="recapture.pin", sep=" ", append=TRUE,
row.names = FALSE, col.names = FALSE)
  #cat("# LogRRDevs \n", file="release.pin", append=TRUE)
  #write.table(matrix(0, ncol=fisheries, nrow=((years/5))), file="release.pin", sep=" ", append=TRUE,
row.names = FALSE, col.names = FALSE)
}

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

if (MEst ==2)
{
  cat("# LogMDevs \n", file="recapture.pin", append=TRUE)
  write.table(t(rep(0, (years/5))), file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col
.names = FALSE)
  cat("# LogMDevs \n", file="release.pin", append=TRUE)
  write.table(t(rep(0, (years/5-1))), file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col
.names = FALSE)
# cat("# LogMDevs \n", file="caa.pin", append=TRUE)
# write.table(t(rep(0, (years/5))), file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names
= FALSE)
}

if (MEst ==3)
{
  cat("# LogMDevs \n", file="recapture.pin", append=TRUE)
  write.table(t(rep(0, (years-1))), file="recapture.pin", sep=" ", append=TRUE, row.names = FALSE, col
.names = FALSE)
  cat("# LogMDevs \n", file="release.pin", append=TRUE)
  write.table(t(rep(0, (years-1))), file="release.pin", sep=" ", append=TRUE, row.names = FALSE, col
.names = FALSE)
# cat("# LogMDevs \n", file="caa.pin", append=TRUE)
# write.table(t(rep(0, (years-1))), file="caa.pin", sep=" ", append=TRUE, row.names = FALSE, col.names
= FALSE)
}

```

## ITCAAN Model Code

```

TOP_OF_MAIN_SECTION
  armblsize = 1000000000; // use instead of gradient_structure::set_ARRAY_MEMBLOCK_SIZE
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(10000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(25000000);

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_CALCS
//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_CALCS

//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_number TrueM //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_CALCS
if (MEst==2){Mlength=yearsby5;
} else if (MEst ==3){ Mlength=yearsm1;
} else{ Mlength=1; }
END_CALCS
!! if (MVaryPhase>0)
init_vector TrueTVM(1,years) //True value for the Time Varying natural mortality only
if it is estimated
init_vector TagLoss(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,stocks) //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
LOCAL_CALCS
    if(RREst==2){RRLength=yearsby5;
    }else if (RREst ==3){ RRLength=yearsml;
    }else{ RRLength=1;    }
END_CALCS
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)
    init_vector TrueRR(1,fisheries)

!!if (RRVaryPhase>0)
    init_matrix TrueTVRR(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_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 test5 //Test value 6
// test to see if the tags never returned have been read in correctly
!!if(test5 != 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
vector TagsRetained(1,stocks) //vector of the probability that a tag remains on a fish at large i
.e. 1-TagLoss
// !!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,agesml,1,fisheries,0.,5.,1) //Selectivity parameters without the
fully selected age
init_bounded_matrix SrvySlctvty(1,agesml,1,regions,0.,5.,1) //Selectivity parameters for the
survey without the fully selected ages which varies by region
init_bounded_matrix LogRecruitmentDevs(2,yearsml,1,stocks,-10.,10.,4) //Recruitment deviation
vector for stock 1 will be put into matrix for calculations
init_bounded_matrix LogEffortDevs(2,years,1,fisheries,-5.,5.,3) //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,regionsml,-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,-10,10,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,yearspl,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
// 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
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)
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 )
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)
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
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
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){
M=TrueM;
}
if (PhaseRR<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);
TagsRetained=1.-TagLoss;
// Movement=TrueMvmnt;
// cout<<"Finished Preliminary Calcs"<<endl;

PROCEDURESECTION
  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])));
        }
    exit(43);
}

} 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]));
    exit(44);
} 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[1]=exp(LogM);
        for (y=1;y<=yearsml;y++)
        {
            M[y+1]=M[y]+exp(LogMDevs(y));
        }
    } 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-1);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();
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];
    for (s=1; s<=stocks; s++)
    {
        //Loop over stock of release
        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) * (TagsRetained(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
        } //End loop over fisheries
        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) * (TagsRetained(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 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];
    for (s=1;s<=stocks;s++)
    {
        //Loop over stock of release
        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)*(TagsRetained(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
        } //End loop over fisheries
    } //End loop over recapture year
} //End loop over tagging 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 loop over stock of release
} //End loop over tagging year
NotReturned=TagsReleased-TotalReturned;
// cout<<"Finished Calculate Tag Returns"<<endl;

```

FUNCTION CalculateObjectiveFunction

```

CatchNLL.initialize(); EffortNLL.initialize(); AgeCompNLL.initialize(); nll.initialize(); TagNLL.
initialize(); SurveyNLL.initialize(); SurveyAgeCompNLL.initialize(); InitAbunNLL.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
CatchNLL=nllNormal(log(column(ObservedCatch,1)),log(column(TotalCatch,1)),exp(LogSigmaCatch(1)));
CatchNLL+=nllNormal(log(column(ObservedCatch,2)),log(column(TotalCatch,2)),exp(LogSigmaCatch(2)));
;
CatchNLL+=nllNormal(log(column(ObservedCatch,3)),log(column(TotalCatch,3)),exp(LogSigmaCatch(3)));
;
CatchNLL+=nllNormal(log(column(ObservedCatch,4)),log(column(TotalCatch,4)),exp(LogSigmaCatch(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
EffortNLL=nllNormal(column(LogEffortDevs,1),zerovec2,exp(LogSigmaEffort(1)));
EffortNLL+=nllNormal(column(LogEffortDevs,2),zerovec2,exp(LogSigmaEffort(2)));
EffortNLL+=nllNormal(column(LogEffortDevs,3),zerovec2,exp(LogSigmaEffort(3)));
EffortNLL+=nllNormal(column(LogEffortDevs,4),zerovec2,exp(LogSigmaEffort(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
        TagNLL=sum(elem_prod(log((TagReturns[ty][s]+myeps)/(TagsReleased(ty,s))),TagsReported[ty
][s]));
    }
    //End loop over stocks
}
//End loop over tag years
TagNLL=sum(elem_prod(log(elem_div(NotReturned+myeps,TagsReleased)),NeverRecovered));
//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+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

REPORTSECTION
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 << mfxp(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(mfxp(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;

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;

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;
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 abundance 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;

if (PhaseRR>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[1]-TrueRR),TrueRR)*100 <<endl;
}

// if (RRVaryPhase>0)
// {
//   myreport<< "#Time Varying Reporting Rate" <<endl;
//   myreport<< ReportingRate <<endl;
//   myreport<< "#Time Varying Reporting Rate True" <<endl;
//   myreport<< TrueTVRR <<endl;
//   myreport<< "#Time Varying Reporting Rate Relative Error" <<endl;
//   myreport<< elem_div((ReportingRate-TrueTVRR),TrueTVRR)*100 <<endl;
// }

if (PhaseM>0)
{
  myreport<< "#Natural Mortality" <<endl;
  myreport<< exp(LogM) <<endl;
  myreport<< "#Natural Mortality True" <<endl;
  myreport<< TrueM <<endl;
  myreport<< "#Natural Mortality Relative Error" <<endl;
  myreport<< ((exp(LogM)-TrueM)/TrueM)*100 <<endl;
}

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

myreport.close();

```