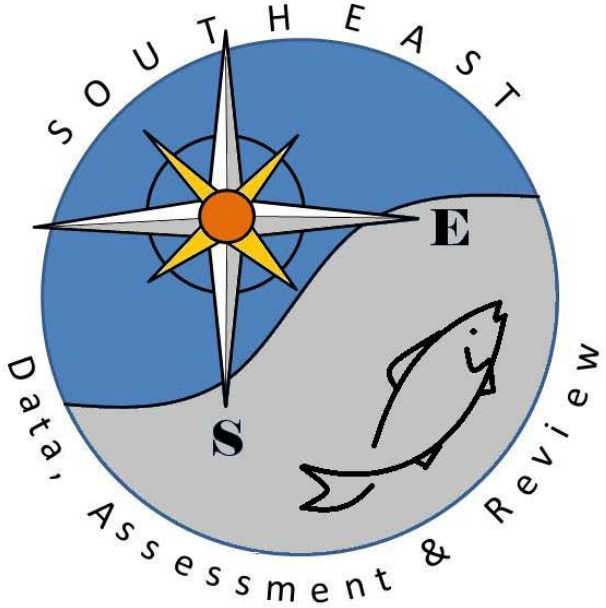
The Beaufort Assessment Model (BAM) with application to cobia: mathematical description, implementation details, and computer code

Kevin Craig

SEDAR28-RW01

15 October 2012



This information is distributed solely for the purpose of peer review. It does not represent and should not be construed to represent any agency determination or policy.

Please cite as:

Craig, K. 2012. The Beaufort Assessment Model (BAM) with application to cobia: mathematical description, implementation details, and computer code. SEDAR28-RW01. SEDAR, North Charleston, SC. 37 pp.

The Beaufort Assessment Model (BAM) with application to cobia: mathematical description, implementation details, and computer code

Sustainable Fisheries Branch

National Marine Fisheries Service

Southeast Fisheries Science Center

NOAA Beaufort Laboratory

101 Pivers Island Road, Beaufort, NC 28516

1 Overview

The primary model in this assessment was the Beaufort assessment model (BAM), which applies a statistical catch-age formulation. The model was implemented with the AD Model Builder software (Fournier et al. 2012), and its structure and equations are detailed herein. In essence, a statistical catch-age model simulates a population forward in time while including fishing processes (Quinn and Deriso 1999; Shertzer et al. 2008). Quantities to be estimated are systematically varied until characteristics of the simulated population match available data on the real population. Statistical catch-age models share many attributes with ADAPT-style tuned and untuned VPAs.

The method of forward projection has a long history in fishery models. It was introduced by Pella and Tomlinson (1969) for fitting production models and then, among many applications, used by Fournier and Archibald (1982), by Deriso et al. (1985) in their CAGEAN model, and by Methot (1989; 2009) in his Stock Synthesis model. The catch-age model of this assessment is similar in structure to the CAGEAN and Stock Synthesis models. Versions of this assessment model have been used in previous SEDAR assessments in the U.S. South Atlantic, such as red porgy, black seabass, snowy grouper, gag grouper, greater amberjack, vermilion snapper, Spanish mackerel, red grouper, red snapper, and tilefish.

2 Model configuration and equations

Model equations are detailed in Table 2.1, and AD Model Builder code is supplied in Appendix A. A general description of the assessment model follows.

Stock dynamics In the assessment model, new biomass was acquired through growth and recruitment, while abundance of existing cohorts experienced exponential decay from fishing and natural mortality. The population was assumed closed to immigration and emigration. The model included age classes $1-12^{+}$, where the oldest age class 12^{+} allowed for the accumulation of fish (i.e., plus group).

Initialization Initial (1950) abundance at age was computed in the model assuming an equilibrium age structure and fishing mortality rate. The equilibrium age structure was computed for ages $1-12^{+}$ based on natural and fishing mortality (F), where F was set equal to the geometric mean fishing mortality from the first three assessment years (1950-1952). This was based on the assumption by the AW panel that the stock was lightly exploited (but less than virgin) prior to the 1950s, particularly during the years following WWII.

Natural mortality rate The natural mortality rate (M) was assumed constant over time, but decreasing with age. The form of M as a function of age was based on Lorenzen (1996). The Lorenzen (1996) approach inversely relates the natural mortality at age to mean weight at age W_a by the power function $M_a=\alpha W_a^{\beta}$, where α is a scale parameter and β is a shape parameter. Lorenzen (1996) provided point estimates of α and β for oceanic fishes, which were used for this assessment. As in previous SEDAR assessments, the Lorenzen estimates of M_a were rescaled to provide the same fraction of fish surviving from age-1 through the oldest observed age (16 yr) as would occur with constant $M = 0.26$ from the DW. This approach using cumulative mortality is consistent with the findings of Hoenig (1983) and Hewitt and Hoenig (2005).

Growth Mean size at age of the population (fork length, FL) was modeled with the von Bertalanffy equation, and weight at age (whole weight, WW) was modeled as a function of fork length. Parameters of growth and conversions (FL-WW) were estimated by the DW and were treated as input to the assessment model. The von Bertalanffy parameter estimates from the DW were $L_\infty = 1324.4$ mm, $k = 0.27$, and $t_0 = -0.47$ yr. For fitting length composition data, the distribution of size at age was assumed normal with coefficient of variation (CV) estimated by the assessment model. A constant CV, rather than constant standard deviation, was suggested by the size at age data.

Female maturity Females were modeled to be fully mature at age 4 and the proportion mature at ages 1, 2, and 3 were estimated to be 0.0, 0.5, and 0.75 respectively.

Spawning stock Spawning stock was modeled using total mature female biomass measured at the time of peak spawning. For cobia, peak spawning was considered to occur in May. In cases when reliable estimates of fecundity are unavailable, spawning biomass is commonly used as a proxy for population fecundity.

Recruitment Expected recruitment of age-1 fish was predicted from spawning stock using the Beverton–Holt spawner-recruit model. Annual variation in recruitment was assumed to occur with lognormal deviations for the years 1975–2009 only. These deviations were constrained to sum to to 1.0 for the period 1984-2009 when annual age compositions and other data sources providing information on year class strength were available. Estimated recruitment deviations for 1975-1983 were not constrained, and provided a bridge between the data poor period beginning in 1950 and the period when age composition data, which contain information on year class strength, became available (1984) (Methot and Taylor 2011). The ending year of estimated recruitment residuals (2009) is based on the age at full selection and the last year of age composition data.

Landings The model included two time series of combined landings plus discards from 1950-2011: a general recreational fleet and a general commercial fleet. Landings were pooled across all gears in the model. Discards were a small proportion of landings and were combined with landings from the respective fleet after applying discard mortality rates provided by the DW. Commercial and recreational discards were assumed negligible prior to 1983 (the first year of regulation).

The combined landings and discards were modeled with the Baranov catch equation (Baranov 1918) and were fitted in units of weight (1000 lb whole weight, commercial) or numbers of fish (1000 fish, recreational). The DW provided observed commercial landings back to the first assessment year (1950). Observed recreational landings were provided by the DW back to 1981 and hindcasts were provided back to 1955. The hindcasting method was extended an additional five years to the start year of the model (1950).

Fishing Mortality For each time series of removals, the assessment model estimated a separate full fishing mortality rate (F). Age-specific rates were then computed as the product of full F and selectivity at age. Apical F was computed as the maximum of F at age summed across fleets.

Selectivities Selectivity curves applied to landings and CPUE series were estimated using a parametric approach. This approach applies plausible structure on the shape of the curves, and achieves greater parsimony than occurs with unique parameters for each age. Selectivity of landings from the commerical and recreational fleets were modeled as flat-topped, using a two parameter logistic function. Selectivities of the fishery dependent indices (Headboat and South Carolina logbook) were assumed the same as that of the general recreational fleet because all use hook and line gear.

Weak priors were used for estimating slope parameters of both selectivity functions. These priors assumed normal distributions with CV = 0.5 (recreational) or CV=0.25 (commercial) and were intended to provide limited information to help the optimization routine during model execution. Starting values for the slope parameters were based on a method of catch curve analysis that simultaneously estimates selectivity and total mortality, while accounting for age-based variation in natural mortality (Thorson and Prager 2011). Priors help by steering estimation away from parameter space with no response in the likelihood surface. Without these priors, it is possible during the optimization search that a selectivity parameter could become unimportant, for example, if its bounds were set too wide and dependent on values of other parameters. When this happens, the likelihood gradient with respect to the aimless parameter approaches zero even if the parameter is not at its globally best value. Diffuse priors help avoid this situation.

Indices of abundance The model was fit to two indices of relative abundance: the Headboat index (1981-2011) and the South Carolina logbook charterboat index (1998-2011). Predicted indices were conditional on selectivities, which were assumed the same for the two indices given that both use hook and line gear, and were computed from abundance at the midpoint of the year.

Catchability In the BAM, catchability scales indices of relative abundance to estimated population abundance at large. Several options for time-varying catchability were implemented in the BAM following recommendations of the 2009 SEDAR procedural workshop on catchability (SEDAR Procedural Guidance 2009). In particular, the BAM

allows for density dependence, linear trends, and random walk, as well as time-invariant catchability. Parameters for these models could be estimated or fixed based on *a priori* considerations. For the base model, the AW assumed time-invariant catchability. For a sensitivity run, however, the AW considered linearly increasing catchability with a slope of 2%, constant after 2003. Choice of the year 2003 was based on recommendations from fishermen regarding when the effects of Global Positioning Systems likely saturated in the southeast U.S. Atlantic (SEDAR 2009). This trend reflects the belief that catchability has generally increased over time as a result of improved technology (SEDAR Procedural Guidance 2009) and as estimated for reef fishes in the Gulf of Mexico (Thorson and Berkson 2010). The value of 2% has been found in other fisheries as well (Zhou et al. 2011). Another sensitivity run applied a random walk approach to estimating catchability. This is notoriously difficult to estimate and often results in the adsorption of noise from the index.

Biological reference points Biological reference points (benchmarks) were calculated based on maximum sustainable yield (MSY) estimates from the Beverton–Holt spawner-recruit model with bias correction (expected values in arithmetic space). Computed benchmarks included MSY, fishing mortality rate at MSY (F_{MSY}), and spawning stock at MSY ($\mathrm{SSB}_{\mathrm{MSY}}$). In this assessment, spawning stock measures total biomass of mature females. These benchmarks are conditional on the estimated selectivity functions and the relative contributions of each fleet's fishing mortality. The selectivity pattern used here was the effort-weighted selectivities at age, with effort from each fishery estimated as the full F averaged over the last three years of the assessment.

Fitting criterion The fitting criterion was a penalized likelihood approach in which observed landings were fit closely, and observed composition data and abundance indices were fit to the degree that they were compatible. Landings and index data were fitted using lognormal likelihoods. Length and age composition data were fitted using robust multinomial likelihoods.

The model includes the capability for each component of the likelihood to be weighted by user-supplied values (for instance, to give more influence to stronger data sources). For data components, these weights were applied by either adjusting CVs (lognormal components) or adjusting effective sample sizes (multinomial components). In this application to cobia, CVs of removals (in arithmetic space) were assumed equal to 0.05, to achieve a close fit to these time series yet allow some imprecision. In practice, the small CVs are a matter of computational convenience, as they help achieve the desired result of close fits to the landings, while avoiding having to solve the Baranov equation iteratively (which is complex when there are multiple fisheries). Weights on other data components (indices, age and length compositions) were adjusted iteratively, starting from initial weights as follows. The CVs of indices were set equal to the values estimated by the DW. Effective sample sizes of the annual length compositions were assumed equal to the annual number of trips sampled. Only number of fish sampled was available for annual age compositions; therefore, effective sample sizes were set to the annual number of fish sampled. Because cobia are caught mostly as one individual fish per trip, the number of fish landed is a probably a good approximation of the number of trips. These initial weights were then adjusted until standard deviations of normalized residuals (SDNRs) were near 1.0 (SEDAR24-RW03, SEDAR25-RW05, Francis 2011). Computed SDNRs accounted for potential correlations in the composition data (TA1.8 in Table A1 of (Francis 2011)). Because only a single pooled age and length composition were available for the commercial fishery, this approach could not be used to derive weights for these data sources. Therefore, weights on commercial age and length compositions were assumed to be the same as those for the comparable recreational data source.

In addition, the compound objective function included several penalties or prior distributions, applied to CV of growth (based on the empirical estimate), the slope of selectivity parameters, and recruitment standard deviation based on Beddington and Cooke (1983) and Mertz and Myers (1996)]. Penalties or priors were applied to maintain parameter estimates near reasonable values, and to prevent the optimization routine from drifting into parameter space with negligible gradient in the likelihood.

Model testing Experiments with a reduced model structure indicated that parameters estimated from the BAM were unbiased and could be recovered from simulated data. Further, the general model structure has been through multiple SEDAR reviews. As an additional measure of quality control, cobia code and input data were examined for accuracy by multiple analysts. This combination of testing and verification procedures suggest that the assessment model is implemented correctly and can provide an accurate assessment of cobia stock dynamics.

References

Baranov, F. I. 1918. On the question of the biological basis of fisheries. Nauchnye Issledovaniya Ikhtiologicheskii Instituta Izvestiya **1**:81–128.

Beddington, J. R., and J. G. Cooke, 1983. The potential yield of fish stocks. FAO Fish. Tech. Pap. 242, 47 p.

Deriso, R. B., T. J. Quinn, and P. R. Neal. 1985. Catch-age analysis with auxiliary information. Canadian Journal of Fisheries and Aquatic Sciences **42**:815–824.

Fournier, D., and C. P. Archibald. 1982. A general theory for analyzing catch at aage data. Canadian Journal of Fisheries and Aquatic Sciences **39**:1195–1207.

Fournier, D. A., H. J. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert. 2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software **27**:233–249.

Francis, R. 2011. Data weighting in statistical fisheries stock assessment models. Canadian Journal of Fisheries and Aquatic Sciences **68**:1124–1138.

Hewitt, D. A., and J. M. Hoenig. 2005. Comparison of two approaches for estimating natural mortality based on longevity. Fishery Bulletin **103**:433–437.

Hoenig, J. M. 1983. Empirical use of longevity data to estimate mortality rates. Fishery Bulletin **81**:898–903.

Lorenzen, K. 1996. The relationship between body weight and natural mortality in juvenile and adult fish: a comparison of natural ecosystems and aquaculture. Journal of Fish Biology **49**:627–642.

Mertz, G., and R. Myers. 1996. Influence of fecundity on recruitment variability of marine fish. Canadian Journal of Fisheries and Aquatic Sciences **53**:1618–1625.

Methot, R., and I. Taylor. 2011. Adjusting for bias due to variability of estimated recruitments in fishery assessment models. Canadian Journal of Fisheries and Aquatic Sciences **68**:1744–1760.

Methot, R. D. 1989. Synthetic estimates of historical abundance and mortality for northern anchovy. American Fisheries Society Symposium **6**:66–82.

Methot, R. D., 2009. User Manual for Stock Synthesis, Model Version 3.04. NOAA Fisheries, Seattle, WA.

Pella, J. J., and P. K. Tomlinson. 1969. A generalized stock production model. Bulletin of the Inter-American Tropical Tuna Commission **13**:419–496.

Quinn, T. J., and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York, New York.

SEDAR, 2009. SEDAR 19: South Atlantic Red Grouper.

SEDAR Procedural Guidance, 2009. SEDAR Procedural Guidance Document 2: Addressing Time-Varying Catchability.

Shertzer, K. W., M. H. Prager, D. S. Vaughan, and E. H. Williams, 2008. Fishery models. Pages 1582–1593 *in* S. E. Jorgensen and F. Fath, editors. Population Dynamics. Vol. [2] of Encyclopedia of Ecology, 5 vols. Elsevier, Oxford.

Thorson, J. T., and J. Berkson. 2010. Multispecies estimation of Bayesian priors for catchability trends and density dependence in the US Gulf of Mexico. Canadian Journal of Fisheries and Aquatic Science **67**:936–954.

Thorson, J. T., and M. Prager. 2011. Better catch curves: Incorporating age-specific natural mortality and logistic selectivity. Transactions of the American Fisheries Society **140**:356–366.

Zhou, S., A. Punt, R. Deng, and B. J. 2011. Estimating multifleet catchability coefficients and natural mortality from fishery catch and effort data: comparison of Bayesian state-space and observation error models. Canadian Journal of Fisheries and Aquatic Science **68**:1171–1181.

Table 2.1. General definitions, input data, population model, and negative log-likelihood components of the statistical catch-age model applied to cobia. Hat notation ($\widehat{*}$) indicates parameters estimated by the assessment model, and breve notation ($\breve{*}$) indicates estimated quantities whose fit to data forms the objective function.

Quantity	Symbol	Description or definition
General Definitions		
Index of years	y	$y \in \{1950 \dots 2011\}$
Index of ages	a	$a \in \{1, 2 \dots A\}$, where $A = 12^{+}$
Index of length bins	l	$l \in \{1, 2 \dots 44\}$
Length bins	l'	$l' \in \{200, 370, \dots, 1970\text{mm}\}$, with midpoint of 30mm bin used to match length compositions. Largest 16 length bins ($FL \geq 1490$ mm) treated as a plus group, but retained for weight calculations.
Index of fisheries	f	$f \in \{1, 2\}$ where 1 = general recreational, 2 = general commercial
Index of CPUE	u	$u \in \{1, 2\}$ where 1 = Headboat, 2 = South Carolina logbook
Input Data		
Observed length compositions	$p^{\lambda}_{(f,u),l,y}$	Proportional contribution of length bin l in year y to fishery f (landings) or index u
Observed age compositions	$p^{\alpha}_{(f,u),a,y}$	Proportional contribution of age class a in year y to fishery f or index u.
Ageing error matrix	$\mathcal{E}$	Estimated from multiple readers ageing the same otoliths.
Length comp. sample sizes	$n^{\lambda}_{(f,u),y}$	Effective number of length samples collected in year y from fishery f or index u
Age comp. sample sizes	$n^{\alpha}_{(f,u),y}$	Effective number of age samples collected in year y from fishery f or index u
Observed landings	$L_{f,y}$	Reported landings in year y from fishery f. Commercial L in 1000 lb whole weight, and recreational L in 1000 fish.
CVs of landings	$c^{L}_{f,y}$	Assumed 0.05 in arithmetic space
Observed abundance indices	$U_{u,y}$	$u = 1$, Headboat (numbers), $y \in \{1981 \dots 2011\}$ $u = 2$, South Carolina logbook (numbers), $y \in \{1998 \dots 2011\}$ Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean.
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1, 2\}$ as above.
Natural mortality rate	M_a	Function of weight at age (w_a): $M_a = \alpha w_a^{\beta}$, with estimates of α and β from Lorenzen (1996). Lorenzen M_a then rescaled based on Hoenig estimate.
Population Model		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages
Proportion females mature at age	m_a	Increasing with age {0, 0.5, 0.75, 1 …,1} for ages $1 - 12^{+}$; assumed constant across years.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Spawning date	t_{spawn}	Fraction denoting the proportional time of year when spawning occurs. Set to 0.42 for cobia by assuming peak spawning occurs in the end of May.
Mean length at age	l_a	Total length (midyear); $l_a = L_\infty(1 - \exp[-K(a - t_0 + 0.5)])$ where K, L_∞, and t_0 are parameters estimated by the DW
CV of l_a	$\widehat{c}_a^{\lambda}$	Estimated coefficient of variation of growth, assumed constant across ages
SD of l_a	σ_a^{λ}	Standard deviation of growth, assumed constant across ages.
Age–length conversion of population	$\psi^u_{a,l}$	$\psi^u_{a,l} = \frac{1}{\sqrt{2\pi}(\sigma_a^{\lambda})} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\sigma_a^{\lambda})^2\right)}$, the Gaussian density function. Matrix ψ^u is rescaled to sum to one within ages, with the largest size a plus group. This matrix is constant across years.
Age–length conversion of landings	$\psi^L_{f,a,l,y}$	$\psi^L_{f,a,l,y} = \psi^u_{a,l}$
Mean length at age of landings	$\xi^L_{(f),a,y}$	Mean length at age from $\psi^L_{f,a,y}$ for landings.
Individual weight at age of population	w_a	Computed from length at age by $w_a = \theta_1 l_a^{\theta_2}$ where θ_1 and θ_2 are parameters from the DW
Individual weight at age of landings	$w^L_{(f),a,y}$	Computed from length at age by $w^L_{(f),a,y} = \theta_1(\xi^L_{(f),a,y})^{\theta_2}$
Fishery and index selectivities	$s_{(f,u),a}$	$s_{(f,u),a} = \frac{1}{1+\exp\left[-\widehat{\eta}_{(f,u)}\left(a - \widehat{\alpha}_{(f,u)}\right)\right]}$ where $\widehat{\eta}_{(f,u)}$ and $\widehat{\alpha}_{(f,u)}$ are estimated parameters. Not all parameters were estimated for each fishery or index; some parameters were fixed as described in the text. For instance, the selectivity of the recreational indices (Headboat and South Carolina logbook) were assumed the same as the general recreational fishery.
Fishing mortality rate of landings	$F_{f,a,y}$	$F_{f,a,y} = s_{f,a,y}\widehat{F}_{f,y}$ where $\widehat{F}_{f,y}$ is an estimated fully selected fishing mortality rate by fishery
Total fishing mortality rate	$F_{a,y}$	$F_{a,y} = \sum_f F_{f,a,y}$
Total mortality rate	$Z_{a,y}$	$Z_{a,y} = M_a + F_{a,y}$
Apical F	F_y	$F_y = \max(F_{a,y})$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Abundance at age	$N_{a,y}$	$N_{1,1950} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}\phi_{init}-0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)\phi_{init}}$ $\widehat{N}_{1+,1950}$ equilibrium conditions expected given assumptions about initial fishing mortality (described below) $N_{1,y+1} = \begin{cases} \frac{0.8\widehat{R}_0\varsigma\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y} & \text{for } y < 1975, \\ \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_{y+1}) & \text{for } y \geq 1975 \end{cases}$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (0 \dots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ $\widehat{R}_0$ (asymptotic maximum recruitment) is an estimated parameter of the spawner-recruit curve, and $\widehat{R}_y$ are estimated annual recruitment deviations in log space for 1975-2009 and zero otherwise. The bias correction is $\varsigma = \exp(\widehat{\sigma_R}^2/2)$, where $\widehat{\sigma_R}^2$ is the estimated variance of recruitment deviations. In the SEDAR-28 baserun, $h = 0.75$ was a fixed parameter. Quantities ϕ_0, ϕ_{init}, and S_y are described below.
Abundance at age (mid-year)	$N'_{a,y}$	Used to match indices of abundance $N'_{a,y} = N_{a,y}\exp(-Z_{a,y}/2)$
Abundance at age at time of spawning	$N''_{a,y}$	Assumed in May to correspond with peak spawning $N''_{a,y} = \exp(-t_{\text{spawn}}Z_{a,y})N_{a,y}$
Unfished abundance at age per recruit at time of spawning	NPR_a	$NPR_0 = 1 \times \exp(-t_{\text{spawn}}M_0)$ $NPR_{a+1} = NPR_a\exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (0 \dots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_A t_{\text{spawn}})]}{1-\exp(-M_A)}$
Initial abundance at age per recruit at time of spawning	NPR_a^{init}	Same calculations as for NPR_a, but including fishing mortality (see Z^{init} below).
Unfished spawning biomass per recruit	ϕ_0	$\phi_0 = \sum_{a=0}^{A} NPR_a\rho_a m_a w_a$ In units of mature female weight.
Initial spawning biomass per recruit	ϕ_{init}	$\phi_{init} = \sum_{a=0}^{A} NPR_a^{init}\rho_a m_a w_a$ In units of mature female weight.
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y}\rho_a m_a w_a$ Spawning biomass is in units of total mature female weight.
Initialization mortality at age	Z_a^{init}	$Z_a^{init} = M_a + s_a^{init}F^{init}$ where F^{init} is an initialization F assumed to be the geometric mean of F from the first three assessment years (1950-1952) and s_a^{init} is the F-weighted average of recreational and commercial selectivity for these three years.
Initial equilibrium abundance at age	N_a^{eq}	Equilibrium age structure given Z_a^{init}
Population biomass	B_y	$B_y = \sum_a N_{a,y}w_a$
Landings at age in numbers	$L'_{f,a,y}$	$L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}}N_{a,y}[1-\exp(-Z_{a,y})]$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Landings at age in whole weight	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y} L'_{f,a,y}$
Index catchability	$q_{u,y}$	$q_{u,1981} = \widehat{q}^0_u f(\text{density})$ $q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ for $y \geq 1981$ Here, $f_y(\text{density}) = (B'_0)^{\widehat{\psi}} (B'_y)^{-\widehat{\psi}}$, where $\widehat{\psi}$ is a parameter to be estimated, $B'_y = \sum_{a=a'}^{A} B_{a,y}$ is annual biomass above some threshold age a', and B'_0 is virgin biomass for ages a' and greater. In practice, a' should be set high enough to give a reasonable summary of exploitable biomass. The function $f(\text{trend})$ provides a model for linear trend (slope of β_q) in catchability from the start of the index until 2003, where technology effects were thought to saturate (see SEDAR 19 DW report). For example, for an index that starts in 1981, $f_y(\text{trend})$ follows, $f_y(\text{trend}) = \begin{cases} 1.0 & :y = 1981 \\ f_{y-1}(\text{trend}) * (\text{y} - 1981)\beta_{\text{q}} & :1981 < y \leq 2003 \\ f_{2003}(\text{trend}) & :2003 < y \end{cases}$ Finally, $f_y(\text{random}) = \exp(\epsilon_{u,y})$ are lognormal catchability deviations which allow for a random walk in catchability when penalties are placed on the $\epsilon_{u,y}$ (see "Objective Function"). In practice, the catchability function $f_y(\text{trend})$ was used as described for the SEDAR-28 cobia assessment. Density dependence and random walks were not applied in the baserun.
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \begin{cases} \sum_a L'_{f,a,y} & :f = 1, \\ \sum_a L''_{f,a,y} & :f = 2 \end{cases}$
Predicted length compositions of fishery independent data	$\breve{p}^{\lambda}_{u,l,y}$	$\breve{p}^{\lambda}_{u,l,y} = \frac{\sum_a \psi_{a,l} s_{u,a,y} N'_{a,y}}{\sum_a s_{u,a,y} N'_{a,y}}$
Predicted length compositions of landings	$\breve{p}^{\lambda}_{f,l,y}$	$\breve{p}^{\lambda}_{f,l,y} = \frac{\sum_a \psi^L_{f,a,l,y} L'_{f,a,y}}{\sum_a L'_{f,a,y}}$
Predicted age compositions	$\breve{p}^{\alpha}_{(f,u),a,y}$	$\breve{p}^{\alpha}_{(f,u),a,y} = \frac{L'_{(f,u),a,y}}{\sum_a L'_{(f,u),a,y}}$
Predicted CPUE	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \widehat{q}_{u,y} \sum_a N'_{a,y} s_{u,a}$ where $s_{u,a}$ is the selectivity of fishery f in the year corresponding to y.

Table 2.1. (continued)

Quantity	Symbol	Description or definition
Objective Function		
Robust multinomial length compositions	Λ_1	$\Lambda_1 = \sum_{f,u}\sum_{y} 0.5\log(E') - \log\left[\exp\left(-\frac{(p^{\lambda}_{(f,u),l,y}-\breve{p}^{\lambda}_{(f,u),l,y})^2}{2E'/(n^{\lambda}_{(f,u),y}\omega^{\lambda}_{(f,u)})}\right)+x\right]$ where $E' = \left[(1-p^{\lambda}_{(f,u),l,y})(p^{\lambda}_{(f,u),l,y})+\frac{0.1}{mbin}\right]$, $mbin$ is the number of length bins, $\omega^{\lambda}_{(f,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. Bins are 30 mm wide.
Robust multinomial age compositions	Λ_2	$\Lambda_2 = \sum_{f,u}\sum_{y} 0.5\log(E') - \log\left[\exp\left(-\frac{(p^{\alpha}_{(f,u),a,y}-\breve{p}^{\alpha}_{(f,u),a,y})^2}{2E'/(n^{\alpha}_{(f,u),y}\omega^{\alpha}_{(f,u)})}\right)+x\right]$ where $E' = \left[(1-p^{\alpha}_{(f,u),a,y})(p^{\alpha}_{(f,u),a,y})+\frac{0.1}{mbin}\right]$, $mbin$ is the number of age bins, $\omega^{\alpha}_{(f,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero.
Lognormal landings	Λ_3	$\Lambda_3 = \sum_{f}\sum_{y} \frac{\left[\log\left((L_{f,y}+x)/(\breve{L}_{f,y}+x)\right)\right]^2}{2(\sigma^L_{f,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^L_{f,y} = \sqrt{\log(1+(c^L_{f,y}/\omega^L_f)^2)}$, with $\omega^L_f = 1$ a preset weight.
Lognormal CPUE	Λ_4	$\Lambda_4 = \sum_{u}\sum_{y} \frac{\left[\log\left((U_{u,y}+x)/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma^U_{u,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma^U_{u,y} = \sqrt{\log(1+(c^U_{u,y}/\omega^U_u)^2)}$, with ω^U_u a preset weight.
Lognormal recruitment deviations	Λ_5	$\Lambda_5 = \omega_5\left[\frac{[R_{1975}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y>1975}^{2009}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\widehat{\sigma}_R)\right]$ where R_y are recruitment deviations in log space, n is the number of years, $\omega_5 = 1$ is a preset weight, $\widehat{\varrho}$ is the first-order autocorrelation, and $\widehat{\sigma}^2_R$ is the estimated recruitment variance ($\varrho = 0$ in the SEDAR-28 base run).
Additional constraint on early recruitment deviations	Λ_6	$\Lambda_6 = \omega_6\left[\frac{[R_{1975}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y=1976}^{Y_1}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\widehat{\sigma}_R)\right]$ where Y_1 is the last year to apply this additional penalty and ω_6 is a preset weight, with ω_6=0.0 for the SEDAR-28 cobia base run.
Additional constraint on final recruitment deviations	Λ_7	$\Lambda_7 = \omega_7\left[\sum_{y=Y_2}^{Y}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where Y_2 is the first year to apply this additional penalty, Y is the terminal year, and ω_7 is a preset weight, with ω_7=0.0 for the SEDAR-28 cobia base run.
Penalty on random walk on catchability	Λ_8	$\Lambda_8 = \omega_8\sum_{u}\sum_{y}\frac{\epsilon^2_{u,y}}{2(\sigma^q_u)^2}$ where ω_8 is a preset weight and σ^q_u is a control variable input by the user defining the standard deviation of the random walk process. As σ^q_u increases, one essentially estimates each deviation as a free parameter, while values close to zero allow little variation in annual catchability. A random walk on catchability was not used for the SEDAR-28 cobia baserun, thus ω_8=0.0.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Penalty on initial age structure	Λ_9	$\Lambda_9 = \sum_{a=1}^{A} (\widehat{N}_{a,1984} - N_a^{eq})^2$ where N_a^{eq} is the equilibrium age structure given the initial F, as defined previously. ω_7=0.0 for the SEDAR-28 cobia base run.
Prior distributions and penalties	Λ_{10}	is the sum of penalty terms used to implement prior distributions on several parameters. Normal priors were applied to $\widehat{\eta}_{(f,u)}$. Normal distributions required a value to describe variance. Normal priors assumed CV=0.5 (i.e., diffuse priors) for $\widehat{\eta}_{(1,u)}$ and CV=0.25 for $\widehat{\eta}_{(2,u)}$.
Apical F penalty	Λ_{11}	$\Lambda_{11} = \begin{cases} 0 & :F_{apex} < 3 \\ \omega_{11} \times \exp^{\sqrt{(F_{apex}-1)}} - 1 & :F_{apex} > 3 \end{cases}$ where $\omega_{11} = 0$ for the SEDAR-28 cobia base run.
Total objective function	Λ	$\Lambda = \sum_{i=1}^{11} \Lambda_i$ Objective function minimized by the assessment model

Appendix A AD Model Builder code to implement the Beaufort Assessment Model

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR 28 Cobia assessment May 2012
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
DATA_SECTION

!!cout << "Starting Beaufort Assessment Model" << endl;
!!cout << endl;
!!cout << "                BAM!" << endl;
!!cout << endl;

// Starting and ending year of the model
init_int styr;
init_int endyr;

//Starting year to estimate recruitment deviation from S-R curve
init_int styr_rec_dev;
//Ending year to estimate recruitment deviation from S-R curve
init_int endyr_rec_dev;
//possible 3 phases of constraints on recruitment deviations
init_int endyr_rec_phase1;
init_int endyr_rec_phase2;

//Ending year for first selectivity period
init_int endyr_period1;

//Total number of ages
init_int nages;

// Vector of ages for age bins
init_vector agebins(1,nages);

//number assessment years
number nyrs;
number nyrs_rec;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   nyrs=endyr-styr+1.;
   nyrs_rec=endyr_rec_dev-styr_rec_dev+1.;
 END_CALCS

//Total number of length bins for each matrix and length bins used to compute mass in largest bin (plus group)
init_int nlenbins;       //used to match data
init_int nlenbins_plus;  //used to compute density of largest bin (plus group)

//Vector of lengths for length bins (mm)(midpoint) and bins used in computation of plus group
init_ivector lenbins(1,nlenbins);
init_ivector lenbins_plus(1,nlenbins_plus);

int nlenbins_all;    //largest size class used to compute average lengths and weights
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   nlenbins_all=nlenbins+nlenbins_plus;
 END_CALCS

//Max F used in spr and msy calcs
init_number max_F_spr_msy;
//Total number of iterations for spr calcs
init_int n_iter_spr;
//Total number of iterations for msy calcs
init_int n_iter_msy;
//Number years at end of time series over which to average sector F's, for weighted selectivities
init_int selpar_n_yrs_wgted;
//bias correction (set to 1.0 for no bias correction or a negative value to compute from rec variance)
init_number set_BiasCor;
//exclude these years from end of time series for computing bias correction
init_number BiasCor_exclude_yrs;

//###################Recreational fishery ########################
//CPUE
init_int styr_mrip_cpue;
init_int endyr_mrip_cpue;
init_vector obs_mrip_cpue(styr_mrip_cpue,endyr_mrip_cpue);//Observed CPUE
init_vector mrip_cpue_cv(styr_mrip_cpue,endyr_mrip_cpue); //CV of cpue

// Landings  (1000 lbs whole weight)
init_int styr_mrip_L;
init_int endyr_mrip_L;
init_vector obs_mrip_L(styr_mrip_L,endyr_mrip_L);   //vector of observed landings by year
init_vector mrip_L_cv(styr_mrip_L,endyr_mrip_L);    //vector of CV of landings by year

// Length Compositions (3 cm bins)
init_int nyr_mrip_lenc;
init_ivector yrs_mrip_lenc(1,nyr_mrip_lenc);
init_vector nsamp_mrip_lenc(1,nyr_mrip_lenc);
init_vector nfish_mrip_lenc(1,nyr_mrip_lenc);
init_matrix obs_mrip_lenc(1,nyr_mrip_lenc,1,nlenbins);

// Age Compositions
init_int nyr_mrip_agec;
init_ivector yrs_mrip_agec(1,nyr_mrip_agec);
init_vector nsamp_mrip_agec(1,nyr_mrip_agec);
init_vector nfish_mrip_agec(1,nyr_mrip_agec);
```

```
init_matrix obs_mrip_agec(1,nyr_mrip_agec,1,nages);

//################Commercial (all sectors) fishery#####################################
//Landings (1000 lbs whole weight)
 init_int styr_cA_L;
 init_int endyr_cA_L;
 init_vector obs_cA_L(styr_cA_L,endyr_cA_L);
 init_vector cA_L_cv(styr_cA_L,endyr_cA_L);

//Length compositions (3 cm bins) of landings
 init_int nyr_cA_lenc;
 init_ivector yrs_cA_lenc(1,nyr_cA_lenc);
 init_vector nsamp_cA_lenc(1,nyr_cA_lenc);
 init_vector nfish_cA_lenc(1,nyr_cA_lenc);
 init_matrix obs_cA_lenc(1,nyr_cA_lenc,1,nlenbins);

 init_int nyr_cA_lenc_pool;   //year and weights to pool predicted comm length comps to match pooled observations
 init_ivector yrs_cA_lenc_pool(1,nyr_cA_lenc_pool);   //years to pool age comps over
 init_vector nsamp_cA_lenc_pool(1,nyr_cA_lenc_pool)

//Age compositions (1 to 12+)
 init_int nyr_cA_agec;
 init_ivector yrs_cA_agec(1,nyr_cA_agec);
 init_vector nsamp_cA_agec(1,nyr_cA_agec);
 init_vector nfish_cA_agec(1,nyr_cA_agec);
 init_matrix obs_cA_agec(1,nyr_cA_agec,1,nages);

 init_int nyr_cA_agec_pool;   //year and weights to pool predicted comm age comps to match pooled observations
 init_ivector yrs_cA_agec_pool(1,nyr_cA_agec_pool);   //years to pool age comps over
 init_vector nsamp_cA_agec_pool(1,nyr_cA_agec_pool);
//###################### HB CPUE data ##############################
//CPUE
 init_int styr_hb_cpue;
 init_int endyr_hb_cpue;
 init_vector obs_hb_cpue(styr_hb_cpue,endyr_hb_cpue);
 init_vector hb_cpue_cv(styr_hb_cpue,endyr_hb_cpue);

//###################### SC logbook CPUE data ##############################
//CPUE
 init_int styr_sc_cpue;
 init_int endyr_sc_cpue;
 init_vector obs_sc_cpue(styr_sc_cpue,endyr_sc_cpue);
 init_vector sc_cpue_cv(styr_sc_cpue,endyr_sc_cpue);

//#################Parameter values and initial guesses ##############################
//#####################################################################################
// Von Bert parameters in TL mm all fish
init_number set_Linf;
init_number set_K;
init_number set_t0;
//Standard errors of von bert params all fish
init_number set_Linf_se;
init_number set_K_se;
init_number set_t0_se;

//CV of length at age and its standard error all fish
init_number set_len_cv;
init_number set_len_cv_se;
//FL(mm)-weight(whole weight in kg) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;
//weight(whole weight)-gonad weight (units=g) relationship: GW=a+b*W; not used here
init_number gwgtpar_a;
init_number gwgtpar_b;

//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age

init_number spawn_time_frac; //time of year of peak spawning, as a fraction of the year

// Natural mortality
init_vector set_M(1,nages);     //age-dependent: used in model
init_number set_M_constant;     //age-independent: used only for MSST and to scale age dependent M, prior if M is estimated
init_number set_M_constant_se;  //SE of age-independent M, used in prior, if M is estimated
init_number max_obs_age;        //max observed age, used to scale M

//Spawner-recruit parameters (Initial guesses or fixed values)
init_number set_steep;         //recruitment steepness
init_number set_steep_se;      //SE of recruitment steepness
init_number set_log_R0;        //recruitment R0
init_number set_R_autocorr;    //recruitment autocorrelation
init_number set_rec_sigma;     //recruitment standard deviation in log space
init_number set_rec_sigma_se;  //SE of recruitment standard deviation in log space

//Initial guesses or fixed values of estimated selectivity parameters
//recreational (all)
init_number set_selpar_L50_mrip;
init_number set_selpar_slope_mrip;
//commercial (all)
init_number set_selpar_L50_cA;
init_number set_selpar_slope_cA;

//--weights for likelihood components-----------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_lc_mrip;
init_number set_w_lc_cA;
init_number set_w_ac_mrip;
init_number set_w_ac_cA;
init_number set_w_I_mrip;
init_number set_w_I_hb;            //weight for headboat index
init_number set_w_I_sc;            //weight for SC charterboat logbook index
init_number set_w_rec;             //for fitting S-R curve
init_number set_w_rec_early;       //additional constraint on early years recruitment
```

```
init_number set_w_rec_end;         //additional constraint on ending years recruitment
init_number set_w_fullF;           //penalty for any Fapex>3(removed in final phase of optimization)
init_number set_w_Ftune;           //weight applied to tuning F (removed in final phase of optimization)
//init_number set_w_cvlen_dev;         //penalty on cv deviations at age
//init_number set_w_cvlen_diff;       //penalty on first difference of cv deviations at age

//////--index catchability------------------------------------------------------------------------------------------------------------
init_number set_logq_mrip;      //catchability coefficient (log) for MRFSS cpue index
init_number set_logq_hb;        //catchability coefficient (log) for headboat cpue index
init_number set_logq_sc;         //catchability coefficient (log) SC charterboat logbook cpue index

//rate of increase on q
init_int set_q_rate_phase;  //value sets estimation phase of rate increase, negative value turns it off
init_number set_q_rate;
//density dependence on fishery q's
init_int set_q_DD_phase;      //value sets estimation phase of random walk, negative value turns it off
init_number set_q_DD_beta;    //value of 0.0 is density independent
init_number set_q_DD_beta_se;
init_int set_q_DD_stage;      //age to begin counting biomass, should be near full exploitation

//random walk on fishery q's
init_int set_q_RW_phase;         //value sets estimation phase of random walk, negative value turns it off
init_number set_q_RW_mrip_var;    //assumed variance of RW q
init_number set_q_RW_hb_var;    //assumed variance of RW q
init_number set_q_RW_sc_var;    //assumed variance of RW q


////--F's--------------------------------
init_number set_log_avg_F_mrip;
init_number set_log_avg_F_cA;

//initial F
init_number set_F_init;        //initial F, scaled by F_init_ratio

//Tune Fapex (tuning removed in final year of optimization)
init_number set_Ftune;
init_int set_Ftune_yr;

//threshold sample sizes for length comps
init_number minSS_mrip_lenc;
init_number minSS_cA_lenc;

//threshold sample sizes for age comps
init_number minSS_mrip_agec;
init_number minSS_cA_agec;

//ageing error matrix (columns are true ages, rows are ages as read for age comps: columns should sum to one)
init_matrix age_error(1,nages,1,nages);

// #######Indexing integers for year(iyear), age(iage),length(ilen) ###############
int iyear;
int iage;
int ilen;
int ff;

number sqrt2pi;
number g2mt;                    //conversion of grams to metric tons
number g2kg;                    //conversion of grams to kg
number g2klb;                   //conversion of grams to 1000 lb
number mt2klb;                  //conversion of metric tons to 1000 lb
number mt2lb;                   //conversion of metric tons to lb
number dzero;                   //small additive constant to prevent division by zero
number huge_number;             //huge number, to avoid irregular parameter space

init_number end_of_data_file;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   if(end_of_data_file!=999)
   {
       cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
       exit(0);  //KWS
   }
   else
   {
    cout << "Data File read correctly" << endl;
   }
 END_CALCS

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
PARAMETER_SECTION
////-------------Growth--------------------------------------------------------------------------

  number Linf;
  //init_bounded_number Linf(800,1700,3);
  number K;
  //init_bounded_number K(0.1,0.7,3);
  number t0;
  //init_bounded_number t0(-1.5,-0.01,3);

  vector meanlen_FL(1,nages);   //mean fork length (mm) at age all fish
  vector wgt_g(1,nages);        //whole wgt in g
  vector wgt_kg(1,nages);       //whole wgt in kg
  vector wgt_mt(1,nages);       //whole wgt in mt
  vector wgt_klb(1,nages);      //whole wgt in 1000 lb
  vector wgt_lb(1,nages);       //whole wgt in lb
  vector gonad_wgt_mt(1,nages); //gonad wgt in mt

  matrix len_mrip_mm(styr,endyr,1,nages);       //mean length at age of mrip landings in mm (may differ from popn mean)
  matrix wholewgt_mrip_klb(styr,endyr,1,nages);      //whole wgt of mrip landings in 1000 lb
  matrix len_cA_mm(styr,endyr,1,nages);             // mean length at age of commercial landings in mm (may differ from popn mean)
  matrix wholewgt_cA_klb(styr,endyr,1,nages);    //whole wgt of commercial landings in 1000 lb
  matrix len_hb_mm(styr,endyr,1,nages);          //not clear if need these for just indices
  matrix wholewgt_hb_klb(styr,endyr,1,nages);     //not clear if need these for just indices
```

```
  matrix len_sc_mm(styr,endyr,1,nages);           //not clear if need these for just indices
  matrix wholewgt_sc_klb(styr,endyr,1,nages);      //not clear if need these for just indices

  matrix lenprob(1,nages,1,nlenbins);           //distn of size at age (age-length key, 3 cm bins) in population
  matrix lenprob_plus(1,nages,1,nlenbins_plus); //used to compute mass in last length bin (a plus group)
  matrix lenprob_all(1,nages,1,nlenbins_all);   //extended lenprob
  vector lenbins_all(1,nlenbins_all);

  //matrices below are used to match length comps
  matrix lenprob_mrip(1,nages,1,nlenbins);     //distn of size at age in mrip
  matrix lenprob_cA(1,nages,1,nlenbins);    //distn of size at age in cA

  //matrices below pertain to the popn at large, used to compute mean weights
  matrix lenprob_mrip_all(1,nages,1,nlenbins_all);     //distn of size at age in mrip
  matrix lenprob_cA_all(1,nages,1,nlenbins_all);    //distn of size at age in crA

  //set min and max equal for constant sd or cv
  init_bounded_number len_cv_val(0.05,0.5,4);
  //number len_cv_val

  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
  //number log_len_cv
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

////----Predicted length and age compositions
  matrix pred_mrip_lenc(1,nyr_mrip_lenc,1,nlenbins);
  matrix pred_cA_lenc(1,nyr_cA_lenc,1,nlenbins);
  matrix pred_mrip_agec(1,nyr_mrip_agec,1,nages);
  matrix ErrorFree_mrip_agec(1,nyr_mrip_agec,1,nages);
  matrix pred_cA_agec(1,nyr_cA_agec,1,nages);
  matrix ErrorFree_cA_agec(1,nyr_cA_agec,1,nages);

  //matrix L_mrip_num_pool(1,nyr_mrip_agec,1,nages);    //landings (numbers) at age pooled for age comps
  //matrix L_mrip_num_pool_yr(1,nyr_mrip_agec_pool,1,nages);  //scaled and weighted landings (numbers) for pooling age comps
  matrix L_cA_num_pool(1,nyr_cA_agec,1,nages);    //landings (numbers) at age pooled for age comps
  matrix L_cA_num_pool_yr(1,nyr_cA_agec_pool,1,nages);  //scaled and weighted landings (numbers) for pooling age comps
  matrix L_cA_num_len_pool(1,nyr_cA_lenc,1,nages);
  matrix L_cA_num_len_pool_yr(1,nyr_cA_lenc_pool,1,nages);  //matrix for weighted length comps (needs to be separate from that for weighting age comps)

//effective sample size applied in multinomial distributions
  vector nsamp_mrip_lenc_allyr(styr,endyr);
  vector nsamp_mrip_agec_allyr(styr,endyr);
  vector nsamp_cA_lenc_allyr(styr,endyr);
  vector nsamp_cA_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_mrip_lenc_allyr(styr,endyr);
  vector nfish_cA_lenc_allyr(styr,endyr)
  vector nfish_mrip_agec_allyr(styr,endyr);
  vector nfish_cA_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_mrip_lenc_allyr_out(styr,endyr);
  vector neff_cA_lenc_allyr_out(styr,endyr);
  vector neff_mrip_agec_allyr_out(styr,endyr);
  vector neff_cA_agec_allyr_out(styr,endyr);

//-----Population---------------------------------------------------------------------------------
  matrix N(styr,endyr+1,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and cpue
  matrix N_spawn(styr,endyr,1,nages);       //Population numbers by year and age at peaking spawning: used for SSB
  vector log_Nage_dev(2,nages);
  vector log_Nage_dev_output(1,nages);      //used in output. equals zero for first age
  matrix B(styr,endyr+1,1,nages);           //Population biomass by year and age at start of yr
  vector totB(styr,endyr+1);                //Total biomass by year
  vector totN(styr,endyr+1);                //Total abundance by year
  vector SSB(styr,endyr);                   //Total spawning biomass by year (total mature female gonad weight)
  vector MatFemB(styr,endyr);               //Total spawning biomass by year (total mature female biomass)
  vector rec(styr,endyr+1);                 //Recruits by year
  vector prop_f(1,nages);                   //Proportion female by age
  vector maturity_f(1,nages);               //Proportion of female mature at age
  vector reprod(1,nages);                   //vector used to compute spawning biomass (total mature female gonad weight)
  vector reprod2(1,nages);                  //vector used to compute mature female biomass

////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(5,18,1);        //log(virgin Recruitment)
  //number log_R0;
  number R0;
                               //virgin recruitment
  //init_bounded_number steep(0.21,0.991,3); //steepness
  number steep;  //uncomment to fix steepness
  init_bounded_number rec_sigma(0.2,1.2,4);  //sd recruitment residuals
  //number rec_sigma;

  number rec_sigma_sq;                       //square of rec_sigma
  number rec_logL_add;                       //additive term in -logL term

 vector log_rec_dev(styr_rec_dev,endyr_rec_dev);
 init_bounded_vector log_rec_dev1(styr_rec_dev,styr_rec_dev+8,-10,10,3);
 init_bounded_dev_vector log_rec_dev2(styr_rec_dev+9,endyr_rec_dev,-10,10,3);

// vector log_rec_dev(styr_rec_dev,endyr_rec_dev)  //create vector to store deviations--not estimated ; initialized to dzero in preliminary_calcs
  vector log_rec_dev_output(styr,endyr+1);           //used in output. equals zero except for yrs in log_rec_dev

  number var_rec_dev;                                //variance of log recruitment deviations, from yrs with unconstrainted S-R(XXXX-XXXX)
  number sigma_rec_dev;                              //sample SD of log residuals (may not equal rec_sigma

  number BiasCor;                                //Bias correction in equilibrium recruits
 // init_bounded_number R_autocorr(-1.0,1.0,-4);  //autocorrelation in SR
  number R_autocorr

  number S0;                                     //equal to spr_F0*R0 = virgin SSB
  number B0;                                     //equal to bpr_F0*R0 = virgin B
```

```
  number R1;                                  //Recruits in styr
  number R_virgin;                            //unfished recruitment with bias correction
  vector SdS0(styr,endyr);                    //SSB / virgin SSB


//-----------------------------------------------------------------------------------------------------------------------------------------------------
////---Selectivity-------------------------------------------------------------------------

//Recreational-------------------------------------------------
  matrix sel_mrip(styr,endyr,1,nages);
  init_bounded_number selpar_L50_mrip(1.0,10.0,2);
  init_bounded_number selpar_slope_mrip(0.5,12.0,2);

//Commercial---------------------------------------------------
  matrix sel_cA(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cA(1.0,10.0,2);
  init_bounded_number selpar_slope_cA(0.5,12.0,2);

//Headboat selectivity
  matrix sel_hb(styr,endyr,1,nages);
//  init_bounded_number selpar_L50_hb(1.0,10.0,2);
//  init_bounded_number selpar_slope_hb(0.5,12.0,2);

//SC logbook selectivity
  matrix sel_sc(styr,endyr,1,nages);
//  init_bounded_number selpar_L50_sc(1.0,10.0,2);
//  init_bounded_number selpar_slope_sc(0.5,12.0,2);

//Weighted total selectivity-----------------------------------------
  //effort-weighted, recent selectivities
  vector sel_wgted_L(1,nages);  //toward landings
  vector sel_wgted_tot(1,nages);//toward Z, landings plus deads discards

//-----------------------------------------------------------------------------------------------------------------------------------------------------
//-------CPUE Predictions--------------------------------
  vector pred_mrip_cpue(styr_mrip_cpue,endyr_mrip_cpue);            //predicted mrip U
  matrix N_mrip(styr_mrip_cpue,endyr_mrip_cpue,1,nages);            //used to compute mrip index
  vector pred_hb_cpue(styr_hb_cpue,endyr_hb_cpue);                  //predicted hb U
  matrix N_hb(styr_hb_cpue,endyr_hb_cpue,1,nages);                  ////used to compute HB index
  vector pred_sc_cpue(styr_sc_cpue,endyr_sc_cpue);                  //predicted sc logbook
  matrix N_sc(styr_sc_cpue,endyr_sc_cpue,1,nages);                  ////used to compute SC index

//---Catchability (CPUE q's)-------------------------------------------------------
  init_bounded_number log_q_mrip(-14,-4,-1);
  init_bounded_number log_q_hb(-14,-4,1);
  init_bounded_number log_q_sc(-14,-4,1);

//  init_bounded_number q_rate(0.001,0.1,set_q_rate_phase);
  number q_rate;
  vector q_rate_fcn_mrip(styr_mrip_cpue,endyr_mrip_cpue);   //increase due to technology creep (saturates in 2003)
  vector q_rate_fcn_hb(styr_hb_cpue,endyr_hb_cpue);         //increase due to technology creep (saturates in 2003)
  vector q_rate_fcn_sc(styr_sc_cpue,endyr_sc_cpue);         //increase due to technology creep (saturates in 2003)


//  init_bounded_number q_DD_beta(0.1,0.9,set_q_DD_phase);
  number q_DD_beta;
  vector q_DD_fcn(styr,endyr);    //density dependent function as a multiple of q (scaled a la Katsukawa and Matsuda. 2003)
  number B0_q_DD;                 //B0 of ages q_DD_age plus
  vector B_q_DD(styr,endyr);      //annual biomass of ages q_DD_age plus

//  init_bounded_vector q_RW_log_dev_mrip(styr_mrip_cpue,endyr_mrip_cpue-1,-3.0,3.0,set_q_RW_phase);
 vector q_RW_log_dev_mrip(styr_mrip_cpue,endyr_mrip_cpue-1);
 vector q_RW_log_dev_hb(styr_hb_cpue,endyr_hb_cpue-1);
 vector q_RW_log_dev_sc(styr_sc_cpue,endyr_sc_cpue-1);


 vector q_mrip(styr_mrip_cpue,endyr_mrip_cpue);
 vector q_hb(styr_hb_cpue,endyr_hb_cpue); //or number q_hb;
 vector q_sc(styr_sc_cpue,endyr_sc_cpue); //or number q_sc;

//-----------------------------------------------------------------------------------------------------------------------------------------------------
//---Landings in numbers (total or 1000 fish) and in wgt (klb)--------------------------------------------------
  matrix L_mrip_num(styr,endyr,1,nages);  //landings (numbers) at age
  matrix L_mrip_klb(styr,endyr,1,nages);  //landings (1000 lb whole weight) at age
  vector pred_mrip_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_mrip_L_klb(styr,endyr);     //yearly landings in 1000 lb summed over ages

  matrix L_cA_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cA_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cA_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cA_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages
  vector obs_cA_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicative bias

  matrix L_total_num(styr,endyr,1,nages);//total landings in number at age
  matrix L_total_klb(styr,endyr,1,nages);//landings in klb at age
  vector L_total_knum_yr(styr,endyr);    //total landings in 1000 fish by yr summed over ages
  vector L_total_klb_yr(styr,endyr);     //total landings (klb) by yr summed over ages

////---MSY calcs-------------------------------------------------------------------------
  number F_mrip_prop;       //proportion of F_sum attributable to hal, last X=selpar_n_yrs_wgted yrs, used for avg body weights
  number F_cA_prop;       //proportion of F_sum attributable to cA, last X yr
  number F_temp_sum;      //sum of geom mean Fsum's in last X yrs, used to compute F_fishery_prop

  vector F_end(1,nages);
  vector F_end_L(1,nages);
  number F_end_apex;

  number SSB_msy_out;           //SSB (total mature biomass) at msy
  number F_msy_out;             //F at msy
  number msy_klb_out;           //max sustainable yield (1000 lb)
  number msy_knum_out;          //max sustainable yield (1000 fish)
  number B_msy_out;             //total biomass at MSY
  number R_msy_out;             //equilibrium recruitment at F=Fmsy
  number spr_msy_out;           //spr at F=Fmsy
```

```
  vector N_age_msy(1,nages);         //numbers at age for MSY calculations: beginning of yr
  vector N_age_msy_mdyr(1,nages);    //numbers at age for MSY calculations: mdpt of yr
  vector L_age_msy(1,nages);         //catch at age for MSY calculations
  vector Z_age_msy(1,nages);         //total mortality at age for MSY calculations
  vector F_L_age_msy(1,nages);       //fishing mortality landings (not discards) at age for MSY calculations
  vector F_msy(1,n_iter_msy);        //values of full F to be used in equilibrium calculations
  vector spr_msy(1,n_iter_msy);      //reproductive capacity-per-recruit values corresponding to F values in F_msy
  vector R_eq(1,n_iter_msy);         //equilibrium recruitment values corresponding to F values in F_msy
  vector L_eq_klb(1,n_iter_msy);     //equilibrium landings(klb) values corresponding to F values in F_msy
  vector L_eq_knum(1,n_iter_msy);    //equilibrium landings(1000 fish) values corresponding to F values in F_msy
  vector SSB_eq(1,n_iter_msy);       //equilibrium reproductive capacity values corresponding to F values in F_msy
  vector B_eq(1,n_iter_msy);         //equilibrium biomass values corresponding to F values in F_msy

  vector FdF_msy(styr,endyr);
  vector SdSSB_msy(styr,endyr);
  number SdSSB_msy_end;
  number FdF_msy_end;
  number FdF_msy_end_mean;          //geometric mean of last 3 yrs

  vector wgt_wgted_L_klb(1,nages);  //fishery-weighted average weight at age of landings
  number wgt_wgted_L_denom;         //used in intermediate calculations

  number iter_inc_msy;              //increments used to compute msy, equals 1/(n_iter_msy-1)

////--------Mortality------------------------------------------------------------------

// Stuff immediately below used only if M is estimated
//  //init_bounded_number M_constant(0.1,0.2,1);                      //age-indpendent: used only for MSST
//  vector Mscale_ages(1,max_obs_age);
//  vector Mscale_len(1,max_obs_age);
//  vector Mscale_wgt_g(1,max_obs_age);
//  vector M_lorenzen(1,max_obs_age);
//  number cum_surv_1plus;

  vector M(1,nages);                        //age-dependent natural mortality
  number M_constant;                        //age-indpendent: used only for MSST

  matrix F(styr,endyr,1,nages);
  vector Fsum(styr,endyr);                  //Full fishing mortality rate by year
  vector Fapex(styr,endyr);                 //Max across ages, fishing mortality rate by year (may differ from Fsum bc of dome-shaped sel
//  sdreport_vector fullF_sd(styr,endyr);
  matrix Z(styr,endyr,1,nages);

  init_bounded_number log_avg_F_mrip(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_mrip(styr_mrip_L,endyr_mrip_L,-12.0,12.0,2);
  matrix F_mrip(styr,endyr,1,nages);
  vector F_mrip_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_mrip;
  number log_F_dev_end_mrip;

  init_bounded_number log_avg_F_cA(-15.0,0.0,1);
  init_bounded_dev_vector log_F_dev_cA(styr_cA_L,endyr_cA_L,-10.0,10.0,2);
  matrix F_cA(styr,endyr,1,nages);
  vector F_cA_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cA;
  number log_F_dev_end_cA;

//  init_bounded_number F_init(0.0001,0.9,4);
//  init_bounded_number F_init(0.01,0.9,-4);
  number F_init

  number F_init_ratio;   //scales initial F, which is read in as a fixed value
  vector sel_initial(1,nages);       //initial selectivity (a combination of for-hire and commercial selectivities)

//---Per-recruit stuff-------------------------------------------------------------------------------
  vector N_age_spr(1,nages);         //numbers at age for SPR calculations: beginning of year
  vector N_age_spr_mdyr(1,nages);    //numbers at age for SPR calculations: midyear
  vector L_age_spr(1,nages);         //catch at age for SPR calculations
  vector Z_age_spr(1,nages);         //total mortality at age for SPR calculations
  vector spr_static(styr,endyr);     //vector of static SPR values by year
  vector F_L_age_spr(1,nages);       //fishing mortality of landings (not discards) at age for SPR calculations
  vector F_spr(1,n_iter_spr);        //values of full F to be used in per-recruit calculations
  vector spr_spr(1,n_iter_spr);      //reproductive capacity-per-recruit values corresponding to F values in F_spr
  vector L_spr(1,n_iter_spr);        //landings(lb)-per-recruit (ypr) values corresponding to F values in F_spr

  vector N_spr_F0(1,nages);          //Used to compute spr at F=0: at time of peak spawning
  vector N_bpr_F0(1,nages);          //Used to compute bpr at F=0: at start of year
  vector N_spr_initial(1,nages);     //Initial spawners per recruit at age given initial F
  vector N_initial_eq(1,nages);      //Initial equilibrium abundance at age
  vector F_initial(1,nages);         //initial F at age
  vector Z_initial(1,nages);         //initial Z at age
  number spr_initial;                //initial spawners per recruit
  number spr_F0;                     //Spawning biomass per recruit at F=0
  number bpr_F0;                     //Biomass per recruit at F=0

  number iter_inc_spr;               //increments used to compute msy, equals max_F_spr_msy/(n_iter_spr-1)


////-------Objective function components---------------------------------------------------------------------
  number w_L;

  number w_lc_mrip;
  number w_lc_cA;

  number w_ac_mrip;
  number w_ac_cA;

  number w_I_mrip;
  number w_I_hb;
  number w_I_sc;

  number w_rec;
  number w_rec_early;
```

```
  number w_rec_end;
  number w_fullF;
  number w_Ftune;
//  number w_cvlen_dev;
//  number w_cvlen_diff;

  number f_mrip_cpue;
  number f_hb_cpue;
  number f_sc_cpue;

  number f_mrip_L;
  number f_cA_L;

  number f_mrip_lenc;
  number f_cA_lenc;

  number f_mrip_agec;
  number f_cA_agec;

  number f_mrip_RW_cpue; //random walk component of indices
  number f_hb_RW_cpue;   //random walk component of indices
  number f_sc_RW_cpue;   //random walk component of indices


  //Penalties and constraints. Not all are used.
  number f_rec_dev;                //weight on recruitment deviations to fit S-R curve
  number f_rec_dev_early;          //extra weight on deviations in first recruitment stanza
  number f_rec_dev_end;            //extra weight on deviations in first recruitment stanza
  number f_rec_historic_dev;       //extra weight on deviations in first recruitment stanza
  number f_Ftune;                  //penalty for tuning F in Ftune yr.  Not applied in final optimization phase.
  number f_fullF_constraint;       //penalty for Fapex>X
//  number f_cvlen_dev_constraint; //deviation penalty on cv's of length at age
//  number f_cvlen_diff_constraint;//first diff penalty on cv's of length at age
  number f_priors;                 //prior information on parameters

  //init_number xdum;
  objective_function_value fval;
  number fval_data;

//--Dummy variables ----
  number denom;                   //denominator used in some calculations
  number numer;                   //numerator used in some calculations

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
INITIALIZATION_SECTION


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
GLOBALS_SECTION
  #include "admodel.h"          // Include AD class definitions
  #include "admb2r.cpp"    // Include S-compatible output functions (needs preceding)

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
RUNTIME_SECTION
 maximum_function_evaluations 1000, 2000,3000, 10000;
 convergence_criteria 1e-2, 1e-2,1e-3, 1e-4;

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
PRELIMINARY_CALCS_SECTION

// Set values of fixed parameters or set initial guess of estimated parameters
  Linf=set_Linf;
  K=set_K;
  t0=set_t0;
  len_cv_val=set_len_cv;

  M=set_M;
  M_constant=set_M_constant;
//  for (iage=1;iage<=max_obs_age;iage++){Mscale_ages(iage)=iage;}

  steep=set_steep;
  R_autocorr=set_R_autocorr;
  rec_sigma=set_rec_sigma;

  log_q_mrip=set_logq_mrip;
  log_q_hb=set_logq_hb;
  log_q_sc=set_logq_sc;

  q_rate=set_q_rate;
  q_rate_fcn_mrip=1.0;
  q_rate_fcn_hb=1.0;
  q_rate_fcn_sc=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;
  q_RW_log_dev_mrip.initialize();
  q_RW_log_dev_hb.initialize();
  q_RW_log_dev_sc.initialize();


  if (set_q_rate_phase<0 & q_rate!=0.0)
  {
      for (iyear=styr_mrip_cpue; iyear<=endyr_mrip_cpue; iyear++)
      {   if (iyear>styr_mrip_cpue & iyear <=2003)
          {//q_rate_fcn_mrip(iyear)=(1.0+q_rate)*q_rate_fcn_mrip(iyear-1); //compound
             q_rate_fcn_mrip(iyear)=(1.0+(iyear-styr_mrip_cpue)*q_rate)*q_rate_fcn_mrip(styr_mrip_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_mrip(iyear)=q_rate_fcn_mrip(iyear-1);}
      }

      for (iyear=styr_hb_cpue; iyear<=endyr_hb_cpue; iyear++)
      {   if (iyear>styr_hb_cpue & iyear <=2003)
```

```
          {//q_rate_fcn_hb(iyear)=(1.0+q_rate)*q_rate_fcn_hb(iyear-1); //compound
             q_rate_fcn_hb(iyear)=(1.0+(iyear-styr_hb_cpue)*q_rate)*q_rate_fcn_hb(styr_hb_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_hb(iyear)=q_rate_fcn_hb(iyear-1);}
      }

      for (iyear=styr_sc_cpue; iyear<=endyr_sc_cpue; iyear++)
      {   if (iyear>styr_sc_cpue & iyear <=2003)
          {//q_rate_fcn_sc(iyear)=(1.0+q_rate)*q_rate_fcn_sc(iyear-1); //compound
             q_rate_fcn_sc(iyear)=(1.0+(iyear-styr_sc_cpue)*q_rate)*q_rate_fcn_sc(styr_sc_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_sc(iyear)=q_rate_fcn_sc(iyear-1);}
      }


  } //end q_rate conditional

  w_L=set_w_L;

  w_lc_mrip=set_w_lc_mrip;
  w_lc_cA=set_w_lc_cA;

  w_ac_mrip=set_w_ac_mrip;
  w_ac_cA=set_w_ac_cA;

  w_I_mrip=set_w_I_mrip;
  w_I_hb=set_w_I_hb;
  w_I_sc=set_w_I_sc;

  w_rec=set_w_rec;
  w_fullF=set_w_fullF;
  w_rec_early=set_w_rec_early;
  w_rec_end=set_w_rec_end;
  w_Ftune=set_w_Ftune;
//  w_cvlen_dev=set_w_cvlen_dev;
//  w_cvlen_diff=set_w_cvlen_diff;

  log_avg_F_mrip=set_log_avg_F_mrip;
  log_avg_F_cA=set_log_avg_F_cA;

  F_init=set_F_init;

  log_R0=set_log_R0;

  selpar_L50_mrip=set_selpar_L50_mrip;
  selpar_slope_mrip=set_selpar_slope_mrip;

  selpar_L50_cA=set_selpar_L50_cA;
  selpar_slope_cA=set_selpar_slope_cA;

  //cout << selpar_L50_cA << selpar_slope_cA << endl;

 sqrt2pi=sqrt(2.*3.14159265);
 g2mt=0.000001;          //conversion of grams to metric tons
 g2kg=0.001;             //conversion of grams to kg
 mt2klb=2.20462;         //conversion of metric tons to 1000 lb
 mt2lb=mt2klb*1000.0;    //conversion of metric tons to lb
 g2klb=g2mt*mt2klb;      //conversion of grams to 1000 lb
 dzero=0.00001;
 huge_number=1.0e+10;

 SSB_msy_out=0.0;

 iter_inc_msy=max_F_spr_msy/(n_iter_msy-1);
 iter_inc_spr=max_F_spr_msy/(n_iter_spr-1);

 maturity_f=maturity_f_obs;
 prop_f=prop_f_obs;

 lenbins_all(1,nlenbins)=lenbins(1,nlenbins);
 for (iyear=1;iyear<=nlenbins_plus; iyear++) {lenbins_all(nlenbins+iyear)=lenbins_plus(iyear);}

//Fill in sample sizes of comps, possibly sampled in nonconsec yrs
//Used primarily for output in R object

      nsamp_mrip_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nsamp_cA_lenc_allyr=missing;
      nsamp_mrip_agec_allyr=missing;
      nsamp_cA_agec_allyr=missing;

      nfish_mrip_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nfish_cA_lenc_allyr=missing;
      nfish_mrip_agec_allyr=missing;
      nfish_cA_agec_allyr=missing;

      for (iyear=1; iyear<=nyr_mrip_lenc; iyear++)
         {if (nsamp_mrip_lenc(iyear)>=minSS_mrip_lenc)
            {nsamp_mrip_lenc_allyr(yrs_mrip_lenc(iyear))=nsamp_mrip_lenc(iyear);
             nfish_mrip_lenc_allyr(yrs_mrip_lenc(iyear))=nfish_mrip_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_mrip_agec; iyear++)
         {if (nsamp_mrip_agec(iyear)>=minSS_mrip_agec)
           {nsamp_mrip_agec_allyr(yrs_mrip_agec(iyear))=nsamp_mrip_agec(iyear);
            nfish_mrip_agec_allyr(yrs_mrip_agec(iyear))=nfish_mrip_agec(iyear);}}

      for (iyear=1; iyear<=nyr_cA_lenc; iyear++)
         {if (nsamp_cA_lenc(iyear)>=minSS_cA_lenc)
           {nsamp_cA_lenc_allyr(yrs_cA_lenc(iyear))=nsamp_cA_lenc(iyear);
            nfish_cA_lenc_allyr(yrs_cA_lenc(iyear))=nfish_cA_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cA_agec; iyear++)
         {if (nsamp_cA_agec(iyear)>=minSS_cA_agec)
           {nsamp_cA_agec_allyr(yrs_cA_agec(iyear))=nsamp_cA_agec(iyear);
            nfish_cA_agec_allyr(yrs_cA_agec(iyear))=nfish_cA_agec(iyear);}}
```

```
//fill in Fs for msy and per-recruit analyses
  F_msy(1)=0.0;
  for (ff=2;ff<=n_iter_msy;ff++)
  {
    F_msy(ff)=F_msy(ff-1)+iter_inc_msy;
  }
  F_spr(1)=0.0;
  for (ff=2;ff<=n_iter_spr;ff++)
  {
    F_spr(ff)=F_spr(ff-1)+iter_inc_spr;
  }


//fill in F's, Catch matrices, and log rec dev with zero's
  F_mrip.initialize();
  L_mrip_num.initialize();
  F_cA.initialize();
  L_cA_num.initialize();

  F_mrip_out.initialize();
  F_cA_out.initialize();

  sel_mrip.initialize();
  sel_cA.initialize();
  sel_hb.initialize();
  sel_sc.initialize();

  log_rec_dev_output.initialize();  //KWS
  //log_rec_historic_dev_output.initialize();
  log_Nage_dev_output.initialize();
  //log_rec_historic_dev.initialize();
  log_rec_dev.initialize();

 //for(iyear=styr_rec_dev;iyear<=endyr_rec_dev; iyear++) {log_rec_dev(iyear)=dzero;}

 //cout << log_rec_dev << endl;

  log_Nage_dev.initialize();


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
TOP_OF_MAIN_SECTION
  arrmblsize=20000000;
  gradient_structure::set_MAX_NVAR_OFFSET(1600);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(2000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(2000000);
  gradient_structure::set_NUM_DEPENDENT_VARIABLES(10000);

//>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
PROCEDURE_SECTION

 RO=mfexp(log_RO);

 //cout<<"start"<<endl;

 //get_M_at_age(); //Needed only if M is estimated

 get_length_weight_at_age();
 //cout << "got length, weight, fecundity transitions" <<endl;
 get_reprod();
 //cout << "got repro stuff" << endl;
 get_length_at_age_dist();
 //cout<< "got predicted length at age distribution"<<endl;
 get_weight_at_age_landings();
 //cout<< "got weight at age of landings"<<endl;
 get_spr_F0();
 //cout << "got F0 spr" << endl;
 get_selectivity();
 //cout << "got selectivity" << endl;
 get_mortality();
 //cout << "got mortalities" << endl;
 get_bias_corr();
 //cout<< "got recruitment bias correction" << endl;
 get_numbers_at_age();
 //cout << "got numbers at age" << endl;
 get_landings_numbers();
 //cout << "got catch at age" << endl;
 get_landings_wgt();
 //cout << "got landings" << endl;
 get_catchability_fcns();
 //cout << "got catchability_fcns" << endl;
 get_indices();
 //cout << "got indices" << endl;
 get_length_comps();
 //cout<< "got length comps"<< endl;
 get_age_comps();
 //cout<< "got age comps"<< endl;
 evaluate_objective_function();
 //cout << "objective function calculations complete" << endl;


FUNCTION get_length_weight_at_age
  //compute mean length (mm FL) and weight (whole) at age
    meanlen_FL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));    //fork length in mm
    wgt_kg=wgtpar_a*pow(meanlen_FL,wgtpar_b);             //wgt in kg
    wgt_g=wgt_kg/g2kg;                                    // convert wgt in kg from L-W relationship to weight in g
    wgt_mt=wgt_g*g2mt;                                        // convert weight in g to weight in mt
//    wgt_g=wgt_mt/g2mt;                                    //wgt in grams
//    wgt_kg=g2kg*wgt_g;                                    //wgt in kilograms
    wgt_klb=mt2klb*wgt_mt;                               //1000 lb of whole wgt
    wgt_lb=mt2lb*wgt_mt;                                 //1000 lb of whole wgt
```

```
    //gonad_wgt_mt=g2mt*(gwgtpar_a+gwgtpar_b*wgt_g_f);    //
    gonad_wgt_mt=g2mt*mfexp(gwgtpar_a+gwgtpar_b*log(wgt_g));    //gonad wgt in mt
    for (iage=1;iage<=nages;iage++)
    {
      if(gonad_wgt_mt(iage)<0){gonad_wgt_mt(iage)=0;}
    }

FUNCTION get_reprod
   //reprod is product of stuff going into reproductive capacity calcs
   reprod=elem_prod(elem_prod(prop_f,maturity_f),wgt_mt);
   reprod2=elem_prod(elem_prod(prop_f,maturity_f),wgt_mt);

FUNCTION get_length_at_age_dist
  //compute matrix of length at age, based on the normal distribution

  for (iage=1;iage<=nages;iage++)
  {
    //len_cv(iage)=mfexp(log_len_cv+log_len_cv_dev(iage));
    len_cv(iage)=len_cv_val;
    len_sd(iage)=meanlen_FL(iage)*len_cv(iage);

    //len_cv(iage)=len_cv_max-(len_cv_max-len_sd)/(1.0+mfexp(-len_cv_slope*(iage-len_cv_a50)));
    for (ilen=1;ilen<=nlenbins_all;ilen++)
     { lenprob_all(iage,ilen)=(mfexp(-(square(lenbins_all(ilen)-meanlen_FL(iage))/
       (2.*square(len_sd(iage)))))/(sqrt2pi*len_sd(iage)));
     }

    lenprob_all(iage)/=sum(lenprob_all(iage)); //standardize to approximate integration and to account for truncated normal (i.e., no sizes<smallest)

    for (ilen=1;ilen<=nlenbins;ilen++) {lenprob(iage,ilen)=lenprob_all(iage,ilen);
    }
    for (ilen=nlenbins+1;ilen<=nlenbins_all;ilen++){lenprob(iage)(nlenbins)=lenprob(iage)(nlenbins)+lenprob_all(iage)(ilen);
    } //plus group
  }
  //fishery specific length probs, assumed normal prior to size limits
  lenprob_mrip=lenprob;
  lenprob_cA=lenprob;

  lenprob_mrip_all=lenprob_all;
  lenprob_cA_all=lenprob_all;

FUNCTION get_weight_at_age_landings

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    len_mrip_mm(iyear)=meanlen_FL;
    wholewgt_mrip_klb(iyear)=wgt_klb;
    len_cA_mm(iyear)=meanlen_FL;
    wholewgt_cA_klb(iyear)=wgt_klb;
    len_hb_mm(iyear)=meanlen_FL;
    wholewgt_hb_klb(iyear)=wgt_klb;
    len_sc_mm(iyear)=meanlen_FL;
    wholewgt_sc_klb(iyear)=wgt_klb;
  }

FUNCTION get_spr_F0
  //at mdyr, apply half this yr's mortality, half next yr's
  N_spr_F0(1)=1.0*mfexp(-1.0*M(1)*spawn_time_frac); //at peak spawning time
  N_bpr_F0(1)=1.0;      //at start of year
  for (iage=2; iage<=nages; iage++)
  {
    //N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(M(iage-1));
    N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(M(iage-1)*(1.0-spawn_time_frac) + M(iage)*spawn_time_frac));
    N_bpr_F0(iage)=N_bpr_F0(iage-1)*mfexp(-1.0*(M(iage-1)));
  }
  N_spr_F0(nages)=N_spr_F0(nages)/(1.0-mfexp(-1.0*M(nages))); //plus group (sum of geometric series)
  N_bpr_F0(nages)=N_bpr_F0(nages)/(1.0-mfexp(-1.0*M(nages)));

  spr_F0=sum(elem_prod(N_spr_F0,reprod));
  bpr_F0=sum(elem_prod(N_bpr_F0,wgt_mt));

FUNCTION get_selectivity

  for (iyear=styr; iyear<=endyr; iyear++)
  {
     sel_mrip(iyear)=logistic(agebins, selpar_L50_mrip, selpar_slope_mrip);
     sel_cA(iyear)=logistic(agebins, selpar_L50_cA, selpar_slope_cA);
//     sel_hb(iyear)=logistic(agebins, selpar_L50_mrip,selpar_slope_mrip);
//     sel_sc(iyear)=logistic(agebins, selpar_L50_mrip,selpar_slope_mrip);
  }
  sel_hb=sel_mrip;
  sel_sc=sel_mrip;

  sel_initial=sel_mrip(styr);

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  //initialization F is avg from first 3 yrs of observed landings
  log_F_dev_init_mrip=sum(log_F_dev_mrip(styr_mrip_L,(styr_mrip_L+2)))/3.0;
  log_F_dev_init_cA=sum(log_F_dev_cA(styr_cA_L,(styr_cA_L+2)))/3.0;

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    //-------------
    if(iyear>=styr_mrip_L & iyear<=endyr_mrip_L)
    {  F_mrip_out(iyear)=mfexp(log_avg_F_mrip+log_F_dev_mrip(iyear)); //}
    // if (iyear<styr_mrip_L){F_mrip_out(iyear)=mfexp(log_avg_F_mrip+log_F_dev_init_mrip);}
       F_mrip(iyear)=sel_mrip(iyear)*F_mrip_out(iyear);
       Fsum(iyear)+=F_mrip_out(iyear);
    }

    if(iyear>=styr_cA_L & iyear<=endyr_cA_L)
```

```
    {  F_cA_out(iyear)=mfexp(log_avg_F_cA+log_F_dev_cA(iyear)); //}
    // if (iyear<styr_cA_L){F_cA_out(iyear)=mfexp(log_avg_F_cA+log_F_dev_init_cA);}
       F_cA(iyear)=sel_cA(iyear)*F_cA_out(iyear);
       Fsum(iyear)+=F_cA_out(iyear);
    }

    //Total F at age
    F(iyear)=F_mrip(iyear); //first in additive series (NO +=)
    F(iyear)+=F_cA(iyear);

    Fapex(iyear)=max(F(iyear));
    Z(iyear)=M+F(iyear);

  }  //end iyear


FUNCTION get_bias_corr
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_dev)-
             sum(log_rec_dev(styr_rec_dev,endyr_rec_dev))/nyrs_rec)
             /(nyrs_rec-1.0);

  //if (set_BiasCor <= 0.0) {BiasCor=mfexp(var_rec_dev/2.0);}   //bias correction
  rec_sigma_sq=square(rec_sigma);
  if (set_BiasCor <= 0.0) {BiasCor=mfexp(rec_sigma_sq/2.0);}   //bias correction
  else {BiasCor=set_BiasCor;}

FUNCTION get_numbers_at_age
//Initialization

  log_rec_dev(styr_rec_dev,styr_rec_dev+8)=log_rec_dev1;
  log_rec_dev(styr_rec_dev+9,endyr_rec_dev)=log_rec_dev2;

  S0=spr_F0*R0;
  R_virgin=(R0/((5.0*steep-1.0)*spr_F0))*
                (BiasCor*4.0*steep*spr_F0-spr_F0*(1.0-steep));
  B0=bpr_F0*R_virgin;
  B0_q_DD=R_virgin*sum(elem_prod(N_bpr_F0(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));


 // F_initial=sel_initial*F_init;
  F_initial=sel_mrip(styr)*mfexp(log_avg_F_mrip+log_F_dev_init_mrip)+
            sel_cA(styr)*mfexp(log_avg_F_cA+log_F_dev_init_cA);

//  F_initial=F_init_ratio*set_F_init;

  Z_initial=M+F_initial;

//Initial equilibrium age structure
  N_spr_initial(1)=1.0*mfexp(-1.0*Z_initial(1)*spawn_time_frac); //at peak spawning time;
  for (iage=2; iage<=nages; iage++)
    {
      N_spr_initial(iage)=N_spr_initial(iage-1)*
                  mfexp(-1.0*(Z_initial(iage-1)*(1.0-spawn_time_frac) + Z_initial(iage)*spawn_time_frac));
    }
  N_spr_initial(nages)=N_spr_initial(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group
//    N_spr_F_init_mdyr(1,(nages-1))=elem_prod(N_spr_initial(1,(nages-1)),
//                                  mfexp((-1.*(M(nages-1)+ F_initial))/2.0));

  spr_initial=sum(elem_prod(N_spr_initial,reprod));

  if (styr==styr_rec_dev) {R1=(R0/((5.0*steep-1.0)*spr_initial))*
                (4.0*steep*spr_initial-spr_F0*(1.0-steep));} //without bias correction (deviation added later)
  else {R1=(R0/((5.0*steep-1.0)*spr_initial))*
                (BiasCor*4.0*steep*spr_initial-spr_F0*(1.0-steep));} //with bias correction

  if(R1<0.0) {R1=1.0;} //Avoid negative popn sizes during search algorithm



//Compute equilibrium age structure for first year
  N_initial_eq(1)=R1;
  for (iage=2; iage<=nages; iage++)
  {
    N_initial_eq(iage)=N_initial_eq(iage-1)*
        mfexp(-1.0*(Z_initial(iage-1)));
  }
  //plus group calculation
  N_initial_eq(nages)=N_initial_eq(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group


//Add deviations to initial equilibrium N
  N(styr)(2,nages)=elem_prod(N_initial_eq(2,nages),mfexp(log_Nage_dev));

  if (styr==styr_rec_dev) {N(styr,1)=N_initial_eq(1)*mfexp(log_rec_dev(styr_rec_dev));}
  else {N(styr,1)=N_initial_eq(1);}

  N_mdyr(styr)(1,nages)=elem_prod(N(styr)(1,nages),(mfexp(-1.*(Z_initial(1,nages))*0.5))); //mid year
  N_spawn(styr)(1,nages)=elem_prod(N(styr)(1,nages),(mfexp(-1.*(Z_initial(1,nages))*spawn_time_frac))); //peak spawning time

  SSB(styr)=sum(elem_prod(N_spawn(styr),reprod));
  MatFemB(styr)=sum(elem_prod(N_spawn(styr),reprod));
  B_q_DD(styr)=sum(elem_prod(N(styr)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));

//Rest of years
  for (iyear=styr; iyear<endyr; iyear++)
  {
    if(iyear<(styr_rec_dev-1)||iyear>endyr_rec_dev) //recruitment follows S-R curve exactly
    {
        //add dzero to avoid log(zero)
        N(iyear+1,1)=BiasCor*mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
            (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero));
        //N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear))*mfexp(log_rec_historic_dev(iyear+1));

        //mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
```

```
        //  (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero)+log_rec_historic_dev(iyear+1));
      N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
      N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
      N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
      N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
      SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
      MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
      B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));

   }
   else   //recruitment follows S-R curve with lognormal deviation
   {
       //add dzero to avoid log(zero)
       N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear))*mfexp(log_rec_dev(iyear+1));

       //mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
           //(1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero)+log_rec_dev(iyear+1));
       N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
       N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
       N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
       N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
       B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
   }
 }


   //last year (projection) has no recruitment variability
   N(endyr+1,1)=SR_func(R0, steep, spr_F0, SSB(endyr));

   //mfexp(log(((0.8*R0*steep*SSB(endyr))/(0.2*R0*spr_F0*
   //             (1.0-steep)+(steep-0.2)*SSB(endyr)))+dzero));
   N(endyr+1)(2,nages)=++elem_prod(N(endyr)(1,nages-1),(mfexp(-1.*Z(endyr)(1,nages-1))));
   N(endyr+1,nages)+=N(endyr,nages)*mfexp(-1.*Z(endyr,nages));//plus group
   //SSB(endyr+1)=sum(elem_prod(N(endyr+1),reprod));


//Time series of interest
  rec=column(N,1);
  SdS0=SSB/S0;

FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_mrip_num(iyear,iage)=N(iyear,iage)*F_mrip(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cA_num(iyear,iage)=N(iyear,iage)*F_cA(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_mrip_L_knum(iyear)=sum(L_mrip_num(iyear))/1000.0;
    pred_cA_L_knum(iyear)=sum(L_cA_num(iyear))/1000.0;
  }


FUNCTION get_landings_wgt

////---Predicted landings------------------------
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    L_mrip_klb(iyear)=elem_prod(L_mrip_num(iyear),wholewgt_mrip_klb(iyear));     //in 1000 lb
    L_cA_klb(iyear)=elem_prod(L_cA_num(iyear),wholewgt_cA_klb(iyear));     //in 1000 lb

    pred_mrip_L_klb(iyear)=sum(L_mrip_klb(iyear));
    pred_cA_L_klb(iyear)=sum(L_cA_klb(iyear));
  }


FUNCTION get_catchability_fcns
 //Get rate increase if estimated, otherwise fixed above
  if (set_q_rate_phase>0.0)
  {
      for (iyear=styr_mrip_cpue; iyear<=endyr_mrip_cpue; iyear++)
      {   if (iyear>styr_mrip_cpue & iyear <=2003)
          {//q_rate_fcn_mrip(iyear)=(1.0+q_rate)*q_rate_fcn_mrip(iyear-1); //compound
             q_rate_fcn_mrip(iyear)=(1.0+(iyear-styr_mrip_cpue)*q_rate)*q_rate_fcn_mrip(styr_mrip_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_mrip(iyear)=q_rate_fcn_mrip(iyear-1);}
      }

      for (iyear=styr_hb_cpue; iyear<=endyr_hb_cpue; iyear++)
      {   if (iyear>styr_hb_cpue & iyear <=2003)
          {//q_rate_fcn_hb(iyear)=(1.0+q_rate)*q_rate_fcn_hb(iyear-1); //compound
             q_rate_fcn_hb(iyear)=(1.0+(iyear-styr_hb_cpue)*q_rate)*q_rate_fcn_hb(styr_hb_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_hb(iyear)=q_rate_fcn_hb(iyear-1);}
      }

      for (iyear=styr_sc_cpue; iyear<=endyr_sc_cpue; iyear++)
      {   if (iyear>styr_sc_cpue & iyear <=2003)
          {//q_rate_fcn_sc(iyear)=(1.0+q_rate)*q_rate_fcn_sc(iyear-1); //compound
             q_rate_fcn_sc(iyear)=(1.0+(iyear-styr_sc_cpue)*q_rate)*q_rate_fcn_sc(styr_sc_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_sc(iyear)=q_rate_fcn_sc(iyear-1);}
      }


  } //end q_rate conditional

 //Get density dependence scalar (=1.0 if density independent model is used)
  if (q_DD_beta>0.0)
  {
```

```
    B_q_DD+=dzero;
    for (iyear=styr;iyear<=endyr;iyear++)
        {q_DD_fcn(iyear)=pow(BO_q_DD,q_DD_beta)*pow(B_q_DD(iyear),-q_DD_beta);}
          //{q_DD_fcn(iyear)=1.0+4.0/(1.0+mfexp(0.75*(B_q_DD(iyear)-0.1*BO_q_DD))); }
 }


FUNCTION get_indices
//---Predicted CPUEs------------------------

 //MRFSS  cpue
  q_mrip(styr_mrip_cpue)=mfexp(log_q_mrip);
  for (iyear=styr_mrip_cpue; iyear<=endyr_mrip_cpue; iyear++)
  {
      //N_mrip(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_mrip(iyear)),wholewgt_mrip_klb(iyear));
      N_mrip(iyear)=elem_prod(N_mdyr(iyear),sel_mrip(iyear));
      pred_mrip_cpue(iyear)=q_mrip(iyear)*q_rate_fcn_mrip(iyear)*q_DD_fcn(iyear)*sum(N_mrip(iyear));
      if (iyear<endyr_mrip_cpue){q_mrip(iyear+1)=q_mrip(iyear)*mfexp(q_RW_log_dev_mrip(iyear));}
 }

 //HB  cpue
  q_hb(styr_hb_cpue)=mfexp(log_q_hb);
  for (iyear=styr_hb_cpue; iyear<=endyr_hb_cpue; iyear++)
  {
      //N_mrip(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_mrip(iyear)),wholewgt_mrip_klb(iyear));
      //use mrip selectivity for headboat
      N_hb(iyear)=elem_prod(N_mdyr(iyear),sel_mrip(iyear));
      pred_hb_cpue(iyear)=q_hb(iyear)*q_rate_fcn_hb(iyear)*q_DD_fcn(iyear)*sum(N_hb(iyear));
      if (iyear<endyr_hb_cpue){q_hb(iyear+1)=q_hb(iyear)*mfexp(q_RW_log_dev_hb(iyear));}
  }

 //SC logbook  cpue
  q_sc(styr_sc_cpue)=mfexp(log_q_sc);
  for (iyear=styr_sc_cpue; iyear<=endyr_sc_cpue; iyear++)
  {
      //N_mrip(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_mrip(iyear)),wholewgt_mrip_klb(iyear));
      //use mrip selectivity for SC charterboat logbook
      N_sc(iyear)=elem_prod(N_mdyr(iyear),sel_mrip(iyear));
      pred_sc_cpue(iyear)=q_sc(iyear)*q_rate_fcn_sc(iyear)*q_DD_fcn(iyear)*sum(N_sc(iyear));
      if (iyear<endyr_sc_cpue){q_sc(iyear+1)=q_sc(iyear)*mfexp(q_RW_log_dev_sc(iyear));}
  }


FUNCTION get_length_comps

 //Recreational
  for (iyear=1;iyear<=nyr_mrip_lenc;iyear++)
  {
        pred_mrip_lenc(iyear)=(L_mrip_num(yrs_mrip_lenc(iyear))*lenprob_mrip)
                        /sum(L_mrip_num(yrs_mrip_lenc(iyear)));
  }

 //All commercial (unweighted)
//  for (iyear=1;iyear<=nyr_cA_lenc;iyear++)
//  {
//        pred_cA_lenc(iyear)=(L_cA_num(yrs_cA_lenc(iyear))*lenprob_cA)
//                          /sum(L_cA_num(yrs_cA_lenc(iyear)));
//  }

// Compute weighted, pooled comm length comps
  L_cA_num_len_pool.initialize();
  for(iyear=1;iyear<=nyr_cA_lenc_pool;iyear++)
  {L_cA_num_len_pool_yr(iyear)=nsamp_cA_lenc_pool(iyear)*L_cA_num(yrs_cA_lenc_pool(iyear))
                           /sum(L_cA_num(yrs_cA_lenc_pool(iyear)));
 //accumulating landings in L_cA_num_pool(1)--correct
  L_cA_num_len_pool(1)+=L_cA_num_len_pool_yr(iyear);}

  for (iyear=1;iyear<=nyr_cA_lenc;iyear++)
          {pred_cA_lenc(iyear)=(L_cA_num_len_pool(iyear)*lenprob_cA)/sum(L_cA_num_len_pool(iyear));}

FUNCTION get_age_comps
//Recreational
//  L_mrip_num_pool.initialize();
//  for(iyear=1;iyear<=nyr_mrip_agec_pool;iyear++)
//  {L_mrip_num_pool_yr(iyear)=nsamp_mrip_agec_pool(iyear)*L_mrip_num(yrs_mrip_agec_pool(iyear))
//                             /sum(L_mrip_num(yrs_mrip_agec_pool(iyear)));
//  L_mrip_num_pool(1)+=L_mrip_num_pool_yr(iyear);}


//  for (iyear=1;iyear<=nyr_mrip_agec;iyear++)
//    {  if (yrs_mrip_agec(iyear)==2006)
///////          {ErrorFree_mrip_agec(iyear)=(L_mrip_num_pool(iyear))/sum(L_mrip_num_pool(iyear));
//           {ErrorFree_mrip_agec(iyear)=(L_mrip_num_pool(iyear))/sum(nsamp_mrip_agec_pool);
//           pred_mrip_agec(iyear)=age_error*(ErrorFree_mrip_agec(iyear)/sum(ErrorFree_mrip_agec(iyear)));}
//       if (yrs_mrip_agec(iyear)>2006)
//          {ErrorFree_mrip_agec(iyear)=(L_mrip_num(yrs_mrip_agec(iyear)))/sum(L_mrip_num(yrs_mrip_agec(iyear)));
//          pred_mrip_agec(iyear)=age_error*ErrorFree_mrip_agec(iyear);}
//    }

 //Recreational
 for (iyear=1;iyear<=nyr_mrip_agec;iyear++)
  {
    ErrorFree_mrip_agec(iyear)=L_mrip_num(yrs_mrip_agec(iyear))/sum(L_mrip_num(yrs_mrip_agec(iyear)));
    pred_mrip_agec(iyear)=age_error*ErrorFree_mrip_agec(iyear);
  }

//Commercial (pool over all years of comm age comp data)
  L_cA_num_pool.initialize();
  for(iyear=1;iyear<=nyr_cA_agec_pool;iyear++)
  {L_cA_num_pool_yr(iyear)=nsamp_cA_agec_pool(iyear)*L_cA_num(yrs_cA_agec_pool(iyear))
                           /sum(L_cA_num(yrs_cA_agec_pool(iyear)));

  L_cA_num_pool(1)+=L_cA_num_pool_yr(iyear);}
```

```
  for (iyear=1;iyear<=nyr_cA_agec;iyear++)
          {ErrorFree_cA_agec(iyear)=(L_cA_num_pool(iyear))/sum(nsamp_cA_agec_pool);
          pred_cA_agec(iyear)=age_error*(ErrorFree_cA_agec(iyear)/sum(ErrorFree_cA_agec(iyear)));}

  //Commercial
  //for (iyear=1;iyear<=nyr_cA_agec;iyear++)
  //{
  //  ErrorFree_cA_agec(iyear)=elem_prod(N(yrs_cA_agec(iyear)),sel_cA(yrs_cA_agec(iyear)));
  //  pred_cA_agec(iyear)=age_error*(ErrorFree_cA_agec(iyear)/sum(ErrorFree_cA_agec(iyear)));
  //}

////--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_mrip+
        sum(log_F_dev_mrip((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cA+
        sum(log_F_dev_cA((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);

  F_mrip_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_mrip+
        sum(log_F_dev_mrip((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cA_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cA+
        sum(log_F_dev_cA((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;

  log_F_dev_end_mrip=sum(log_F_dev_mrip((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_cA=sum(log_F_dev_cA((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

  F_end_L=sel_mrip(endyr)*mfexp(log_avg_F_mrip+log_F_dev_end_mrip)+
          sel_cA(endyr)*mfexp(log_avg_F_cA+log_F_dev_end_cA);

  F_end=F_end_L;
  F_end_apex=max(F_end);

  sel_wgted_tot=F_end/F_end_apex;
  sel_wgted_L=elem_prod(sel_wgted_tot, elem_div(F_end_L,F_end));

  wgt_wgted_L_denom=F_mrip_prop+F_cA_prop;
  wgt_wgted_L_klb=F_mrip_prop/wgt_wgted_L_denom*wholewgt_mrip_klb(endyr)+
                  F_cA_prop/wgt_wgted_L_denom*wholewgt_cA_klb(endyr);

FUNCTION get_msy

  //compute values as functions of F
  for(ff=1; ff<=n_iter_msy; ff++)
  {
    //uses fishery-weighted F's
    Z_age_msy=0.0;
    F_L_age_msy=0.0;

    F_L_age_msy=F_msy(ff)*sel_wgted_L;
    Z_age_msy=M+F_L_age_msy;

    N_age_msy(1)=1.0;
    for (iage=2; iage<=nages; iage++)
    {
      N_age_msy(iage)=N_age_msy(iage-1)*mfexp(-1.*Z_age_msy(iage-1));
    }
    N_age_msy(nages)=N_age_msy(nages)/(1.0-mfexp(-1.*Z_age_msy(nages)));
    N_age_msy_mdyr(1,(nages-1))=elem_prod(N_age_msy(1,(nages-1)),
                                mfexp((-1.*Z_age_msy(1,(nages-1)))*spawn_time_frac));
    N_age_msy_mdyr(nages)=(N_age_msy_mdyr(nages-1)*
                           (mfexp(-1.*(Z_age_msy(nages-1)*(1.0-spawn_time_frac) +
                                 Z_age_msy(nages)*spawn_time_frac) )))
                           /(1.0-mfexp(-1.*Z_age_msy(nages)));

    spr_msy(ff)=sum(elem_prod(N_age_msy_mdyr,reprod));


    //Compute equilibrium values of R (including bias correction), SSB and Yield at each F
    R_eq(ff)=(RO/((5.0*steep-1.0)*spr_msy(ff)))*
             (BiasCor*4.0*steep*spr_msy(ff)-spr_F0*(1.0-steep));
    if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
    N_age_msy*=R_eq(ff);
    N_age_msy_mdyr*=R_eq(ff);

    for (iage=1; iage<=nages; iage++)
    {
      L_age_msy(iage)=N_age_msy(iage)*(F_L_age_msy(iage)/Z_age_msy(iage))*
                      (1.-mfexp(-1.*Z_age_msy(iage)));
    }

    SSB_eq(ff)=sum(elem_prod(N_age_msy_mdyr,reprod));
    B_eq(ff)=sum(elem_prod(N_age_msy,wgt_mt));
    L_eq_klb(ff)=sum(elem_prod(L_age_msy,wgt_wgted_L_klb));
    L_eq_knum(ff)=sum(L_age_msy)/1000.0;
  }

  msy_klb_out=max(L_eq_klb);

  for(ff=1; ff<=n_iter_msy; ff++)
  {
   if(L_eq_klb(ff) == msy_klb_out)
      {
        SSB_msy_out=SSB_eq(ff);
        B_msy_out=B_eq(ff);
        R_msy_out=R_eq(ff);
        msy_knum_out=L_eq_knum(ff);
        F_msy_out=F_msy(ff);
        spr_msy_out=spr_msy(ff);
      }
  }

//--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
FUNCTION get_miscellaneous_stuff

//switch here if var_rec_dev <=dzero
  if(var_rec_dev>0.0)
   {sigma_rec_dev=sqrt(var_rec_dev);} //pow(var_rec_dev,0.5);  //sample SD of predicted residuals (may not equal rec_sigma)
   else{sigma_rec_dev=0.0;}

  len_cv=elem_div(len_sd,meanlen_FL);

  //compute total landings- and discards-at-age in 1000 fish and klb
  L_total_num.initialize();
  L_total_klb.initialize();
  L_total_knum_yr.initialize();
  L_total_klb_yr.initialize();

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_klb_yr(iyear)=pred_mrip_L_klb(iyear)+pred_cA_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_mrip_L_knum(iyear)+pred_cA_L_knum(iyear);

        B(iyear)=elem_prod(N(iyear),wgt_mt);
        totN(iyear)=sum(N(iyear));
        totB(iyear)=sum(B(iyear));
  }

  L_total_num=L_mrip_num+L_cA_num;   //landings at age in number fish
  L_total_klb=L_mrip_klb+L_cA_klb;   //landings at age in klb whole weight

  B(endyr+1)=elem_prod(N(endyr+1),wgt_mt);
  totN(endyr+1)=sum(N(endyr+1));
  totB(endyr+1)=sum(B(endyr+1));

//  steep_sd=steep;
//  fullF_sd=Fsum;

  if(F_msy_out>0)
    {
      FdF_msy=Fapex/F_msy_out;
      FdF_msy_end=FdF_msy(endyr);
      FdF_msy_end_mean=pow((FdF_msy(endyr)*FdF_msy(endyr-1)*FdF_msy(endyr-2)),(1.0/3.0));
    }
  if(SSB_msy_out>0)
    {
      SdSSB_msy=SSB/SSB_msy_out;
      SdSSB_msy_end=SdSSB_msy(endyr);
    }

   //fill in log recruitment deviations for yrs they are nonzero
   for(iyear=styr_rec_dev; iyear<=endyr_rec_dev; iyear++)
     {log_rec_dev_output(iyear)=log_rec_dev(iyear);}
   //for(iyear=(styr+1); iyear<=(styr_rec_dev-1); iyear++)
   //  {log_rec_historic_dev_output(iyear)=log_rec_historic_dev(iyear);}
   //fill in log Nage deviations for ages they are nonzero (ages2+)
   for(iage=2; iage<=nages; iage++)
   {
    log_Nage_dev_output(iage)=log_Nage_dev(iage);
   }


//--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_per_recruit_stuff

  //static per-recruit stuff

  for(iyear=styr; iyear<=endyr; iyear++)
  {
    N_age_spr(1)=1.0;
    for(iage=2; iage<=nages; iage++)
    {
      N_age_spr(iage)=N_age_spr(iage-1)*mfexp(-1.*Z(iyear,iage-1));
    }
    N_age_spr(nages)=N_age_spr(nages)/(1.0-mfexp(-1.*Z(iyear,nages)));
    N_age_spr_mdyr(1,(nages-1))=elem_prod(N_age_spr(1,(nages-1)),
                               mfexp(-1.*Z(iyear)(1,(nages-1))*spawn_time_frac));
    N_age_spr_mdyr(nages)=(N_age_spr_mdyr(nages-1)*
                          (mfexp(-1.*(Z(iyear)(nages-1)*(1.0-spawn_time_frac) + Z(iyear)(nages)*spawn_time_frac) )))
                          /(1.0-mfexp(-1.*Z(iyear)(nages)));
    spr_static(iyear)=sum(elem_prod(N_age_spr_mdyr,reprod))/spr_F0;
  }


  //compute SSB/R and YPR as functions of F
  for(ff=1; ff<=n_iter_spr; ff++)
  {
    //uses fishery-weighted F's, same as in MSY calculations
    Z_age_spr=0.0;
    F_L_age_spr=0.0;

    F_L_age_spr=F_spr(ff)*sel_wgted_L;

    Z_age_spr=M+F_L_age_spr;

    N_age_spr(1)=1.0;
    for (iage=2; iage<=nages; iage++)
    {
      N_age_spr(iage)=N_age_spr(iage-1)*mfexp(-1.*Z_age_spr(iage-1));
    }
    N_age_spr(nages)=N_age_spr(nages)/(1-mfexp(-1.*Z_age_spr(nages)));
    N_age_spr_mdyr(1,(nages-1))=elem_prod(N_age_spr(1,(nages-1)),
                                mfexp((-1.*Z_age_spr(1,(nages-1)))*spawn_time_frac));
    N_age_spr_mdyr(nages)=(N_age_spr_mdyr(nages-1)*
                          (mfexp(-1.*(Z_age_spr(nages-1)*(1.0-spawn_time_frac) + Z_age_spr(nages)*spawn_time_frac) )))
                          /(1.0-mfexp(-1.*Z_age_spr(nages)));
```

```
    spr_spr(ff)=sum(elem_prod(N_age_spr_mdyr,reprod));
    L_spr(ff)=0.0;
    for (iage=1; iage<=nages; iage++)
    {
      L_age_spr(iage)=N_age_spr(iage)*(F_L_age_spr(iage)/Z_age_spr(iage))*
                      (1.-mfexp(-1.*Z_age_spr(iage)));
      L_spr(ff)+=L_age_spr(iage)*wgt_wgted_L_klb(iage)*1000.0; //in lb
    }
  }

FUNCTION get_effective_sample_sizes

      neff_mrip_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
      neff_cA_lenc_allyr_out=missing;
      neff_mrip_agec_allyr_out=missing;
      neff_cA_agec_allyr_out=missing;

      for (iyear=1; iyear<=nyr_mrip_lenc; iyear++)
         {if (nsamp_mrip_lenc(iyear)>=minSS_mrip_lenc)
            {neff_mrip_lenc_allyr_out(yrs_mrip_lenc(iyear))=multinom_eff_N(pred_mrip_lenc(iyear),obs_mrip_lenc(iyear));}
          else {neff_mrip_lenc_allyr_out(yrs_mrip_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cA_lenc; iyear++)
         {if (nsamp_cA_lenc(iyear)>=minSS_cA_lenc)
            {neff_cA_lenc_allyr_out(yrs_cA_lenc(iyear))=multinom_eff_N(pred_cA_lenc(iyear),obs_cA_lenc(iyear));}
          else {neff_cA_lenc_allyr_out(yrs_cA_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_mrip_agec; iyear++)
         {if (nsamp_mrip_agec(iyear)>=minSS_mrip_agec)
            {neff_mrip_agec_allyr_out(yrs_mrip_agec(iyear))=multinom_eff_N(pred_mrip_agec(iyear),obs_mrip_agec(iyear));}
          else {neff_mrip_agec_allyr_out(yrs_mrip_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cA_agec; iyear++)
         {if (nsamp_cA_agec(iyear)>=minSS_cA_agec)
            {neff_cA_agec_allyr_out(yrs_cA_agec(iyear))=multinom_eff_N(pred_cA_agec(iyear),obs_cA_agec(iyear));}
          else {neff_cA_agec_allyr_out(yrs_cA_agec(iyear))=-99;}
         }
//--------------------------------------------------------------------------------------------------------------------------------------------------------------


FUNCTION evaluate_objective_function
  fval=0.0;
  fval_data=0.0;
//---likelihoods---------------------------

//---Indices-------------------------------

  f_mrip_cpue=0.0;
  //f_mrip_cpue=lk_lognormal(pred_mrip_cpue, obs_mrip_cpue, mrip_cpue_cv, w_I_mrip);
  //fval+=f_mrip_cpue;
  //fval_data+=f_mrip_cpue;

  f_hb_cpue=0.0;
  f_hb_cpue=lk_lognormal(pred_hb_cpue, obs_hb_cpue, hb_cpue_cv, w_I_hb);
  fval+=f_hb_cpue;
  fval_data+=f_hb_cpue;

  f_sc_cpue=0.0;
  f_sc_cpue=lk_lognormal(pred_sc_cpue, obs_sc_cpue, sc_cpue_cv, w_I_sc);
  fval+=f_sc_cpue;
  fval_data+=f_sc_cpue;

////---Landings-------------------------------

  //f_mrip_L in 1000 fish
  f_mrip_L=lk_lognormal(pred_mrip_L_knum(styr_mrip_L,endyr_mrip_L), obs_mrip_L(styr_mrip_L,endyr_mrip_L),
                        mrip_L_cv(styr_mrip_L,endyr_mrip_L), w_L);

 //f_mrip_L in 1000 lb ww KC
 // f_mrip_L=lk_lognormal(pred_mrip_L_klb(styr_mrip_L,endyr_mrip_L), obs_mrip_L(styr_mrip_L,endyr_mrip_L),
 //                       mrip_L_cv(styr_mrip_L,endyr_mrip_L), w_L);
  fval+=f_mrip_L;
  fval_data+=f_mrip_L;

  //f_cA_L in 1000 lb wholewgt
  f_cA_L=lk_lognormal(pred_cA_L_klb(styr_cA_L,endyr_cA_L), obs_cA_L(styr_cA_L,endyr_cA_L),
                      cA_L_cv(styr_cA_L,endyr_cA_L), w_L);
  fval+=f_cA_L;
  fval_data+=f_cA_L;

//---Length comps--------------------------------

  //f_mrip_lenc
  f_mrip_lenc=lk_robust_multinomial(nsamp_mrip_lenc, pred_mrip_lenc, obs_mrip_lenc, nyr_mrip_lenc, double(nlenbins), minSS_mrip_lenc, w_lc_mrip);
  //f_mrip_lenc=lk_multinomial(nsamp_mrip_lenc, pred_mrip_lenc, obs_mrip_lenc, nyr_mrip_lenc, minSS_mrip_lenc, w_lc_mrip);
  fval+=f_mrip_lenc;
  fval_data+=f_mrip_lenc;

  //f_cA_lenc
  f_cA_lenc=lk_robust_multinomial(nsamp_cA_lenc, pred_cA_lenc, obs_cA_lenc, nyr_cA_lenc, double(nlenbins), minSS_cA_lenc, w_lc_cA);
  //f_cA_lenc=lk_multinomial(nsamp_cA_lenc, pred_cA_lenc, obs_cA_lenc, nyr_cA_lenc, minSS_cA_lenc, w_lc_cA);
  fval+=f_cA_lenc;
  fval_data+=f_cA_lenc;

//////---Age comps--------------------------------

  //f_mrip_agec
  f_mrip_agec=lk_robust_multinomial(nsamp_mrip_agec, pred_mrip_agec, obs_mrip_agec, nyr_mrip_agec, double(nages), minSS_mrip_agec, w_ac_mrip);
  //f_mrip_agec=lk_multinomial(nsamp_mrip_agec, pred_mrip_agec, obs_mrip_agec, nyr_mrip_agec, minSS_mrip_agec, w_ac_mrip);
  fval+=f_mrip_agec;
  fval_data+=f_mrip_agec;
```

```
  //f_cA_agec
  f_cA_agec=lk_robust_multinomial(nsamp_cA_agec, pred_cA_agec, obs_cA_agec, nyr_cA_agec, double(nages), minSS_cA_agec, w_ac_cA);
  //f_cA_agec=lk_multinomial(nsamp_cA_agec, pred_cA_agec, obs_cA_agec, nyr_cA_agec, minSS_cA_agec, w_ac_cA);
  fval+=f_cA_agec;
  fval_data+=f_cA_agec;

////-----------Constraints and penalties-------------------------------
  f_rec_dev=0.0;
  //rec_sigma_sq=square(rec_sigma);
  rec_logL_add=nyrs_rec*log(rec_sigma);
  f_rec_dev=(square(log_rec_dev(styr_rec_dev) + rec_sigma_sq/2.0)/(2.0*rec_sigma_sq));
  for(iyear=(styr_rec_dev+1); iyear<=endyr; iyear++)
  {f_rec_dev+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
              (2.0*rec_sigma_sq));}
  f_rec_dev+=rec_logL_add;
  fval+=w_rec*f_rec_dev;

  f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
  if (w_rec_early>0.0)
    { if (styr_rec_dev<endyr_rec_phase1)
        {
          for(iyear=styr_rec_dev; iyear<=endyr_rec_phase1; iyear++)
          //{f_rec_dev_early+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
          //                  (2.0*rec_sigma_sq)) + rec_logL_add;}
          {f_rec_dev_early+=square(log_rec_dev(iyear));}
        }
  fval+=w_rec_early*f_rec_dev_early;
  }

  f_rec_dev_end=0.0; //possible extra constraint on ending rec deviations
  if (w_rec_end>0.0)
  { if (endyr_rec_phase2<endyr)
        {
          for(iyear=(endyr_rec_phase2+1); iyear<=endyr; iyear++)
          //{f_rec_dev_end+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
          //                (2.0*rec_sigma_sq)) + rec_logL_add;}
          {f_rec_dev_end+=square(log_rec_dev(iyear));}
        }
      fval+=w_rec_end*f_rec_dev_end;
   }


  //fval+=norm2(log_Nage_dev); //applies if initial age structure is estimated


  //Random walk components of fishery dependent indices
  f_mrip_RW_cpue=0.0;
  for (iyear=styr_mrip_cpue; iyear<endyr_mrip_cpue; iyear++)
      {f_mrip_RW_cpue+=square(q_RW_log_dev_mrip(iyear))/(2.0*set_q_RW_mrip_var);}
  fval+=f_mrip_RW_cpue;

  f_hb_RW_cpue=0.0;
  for (iyear=styr_hb_cpue; iyear<endyr_hb_cpue; iyear++)
      {f_hb_RW_cpue+=square(q_RW_log_dev_hb(iyear))/(2.0*set_q_RW_hb_var);}
  fval+=f_hb_RW_cpue;

  f_sc_RW_cpue=0.0;
  for (iyear=styr_sc_cpue; iyear<endyr_sc_cpue; iyear++)
      {f_sc_RW_cpue+=square(q_RW_log_dev_sc(iyear))/(2.0*set_q_RW_sc_var);}
  fval+=f_sc_RW_cpue;


//---Priors-----------------------------------------------------
//neg_log_prior arguments: estimate, prior, variance, pdf type
//Variance input as a negative value is considered to be CV in arithmetic space (CV=-1 implies loose prior)
//pdf type 1=none, 2=lognormal, 3=normal, 4=beta
  f_priors=0.0;
  //f_priors+=neg_log_prior(len_cv_val, set_len_cv, square(set_len_cv_se), 3);
  //f_priors+=neg_log_prior(Linf,set_Linf,square(set_Linf_se),3);
  //f_priors+=neg_log_prior(K,set_K,square(set_K_se),3);
  //f_priors+=neg_log_prior(t0,set_K,square(set_t0_se),3);

  //f_priors+=neg_log_prior(steep, set_steep, square(set_steep_se), 4);
  //f_priors+=neg_log_prior(rec_sigma,set_rec_sigma,square(set_rec_sigma_se),3);
  //f_priors+=neg_log_prior(rec_sigma,set_rec_sigma,0.5,3);
  //f_priors+=neg_log_prior(R_autocorr,set_R_autocorr, 1.0, 1);
  //f_priors+=neg_log_prior(q_DD_beta, set_q_DD_beta, square(set_q_DD_beta_se), 2);
  //f_priors+=neg_log_prior(q_rate, set_q_rate, dzero+square(set_q_rate), 2);
  //f_priors+=neg_log_prior(F_init, set_F_init, -1.0 , 2);
  //f_priors+=neg_log_prior(M_constant, set_M_constant, square(set_M_constant_se), 2);

  //f_priors+=neg_log_prior(selpar_L50_mrip, set_selpar_L50_mrip, -0.5, 3);
  f_priors+=neg_log_prior(selpar_slope_mrip,set_selpar_slope_mrip, -0.5, 3);
  //f_priors+=neg_log_prior(selpar_L50_cA, set_selpar_L50_cA, -0.25, 3);
  f_priors+=neg_log_prior(selpar_slope_cA,set_selpar_slope_cA, -0.25, 3);

  //f_priors+=neg_log_prior(steep,set_steep,-0.5,3);  //weak prior on steepness

  fval+=f_priors;

//--------------------------------------------------------------------------------
//Logistic function: 2 parameters
FUNCTION dvar_vector logistic(const dvar_vector& ages, const dvariable& L50, const dvariable& slope)
  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1./(1.+mfexp(-1.*slope*(ages-L50))); //logistic;
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

//--------------------------------------------------------------------------------
//Logistic function: 4 parameters
FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
```

```
  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase, L502=age at 50% decrease additive to L501, slope2=slope of decrease
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=elem_prod( (1./(1.+mfexp(-1.*slope1*(ages-L501)))),(1.-(1./(1.+mfexp(-1.*slope2*(ages-(L501+L502))))))  );
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

//--------------------------------------------------------------------------------
//Jointed logistic function: 6 parameters (increasing and decreasing logistics joined at peak selectivity)
FUNCTION dvar_vector logistic_joint(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)
  //ages=vector of ages, L501=age at 50% sel (ascending limb), slope1=rate of increase,L502=age at 50% sel (descending), slope1=rate of increase (ascending),
  //satval=saturation value of descending limb, joint=location in age vector to join curves (may equal age or age + 1 if age-0 is included)
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1.0;
  for (iage=1; iage<=nages; iage++)
  {
   if (double(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope1*(ages(iage)-L501)));}
   if (double(iage)>joint){Sel_Tmp(iage)=1.0-(1.0-satval)/(1.+mfexp(-1.*slope2*(ages(iage)-L502)));}
  }
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

//--------------------------------------------------------------------------------
//Double Gaussian function: 6 parameters (as in SS3)
FUNCTION dvar_vector gaussian_double(const dvar_vector& ages, const dvariable& peak, const dvariable& top, const dvariable& ascwid, const dvariable& deswid, const dvariable& init, const dvariable& final)
  //ages=vector of ages, peak=ascending inflection location (as logistic), top=width of plateau, ascwid=ascent width (as log(width))
  //deswid=descent width (as log(width))
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step1(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step2(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step3(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step4(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step5(ages.indexmin(),ages.indexmax());
  dvar_vector sel_step6(ages.indexmin(),ages.indexmax());
  dvar_vector pars_tmp(1,6); dvar_vector sel_tmp_iq(1,2);

  pars_tmp(1)=peak;
  pars_tmp(2)=peak+1.0+(0.99*ages(nages)-peak-1.0)/(1.0+mfexp(-top));
  pars_tmp(3)=mfexp(ascwid);
  pars_tmp(4)=mfexp(deswid);
  pars_tmp(5)=1.0/(1.0+mfexp(-init));
  pars_tmp(6)=1.0/(1.0+mfexp(-final));

  sel_tmp_iq(1)=mfexp(-(square(ages(1)-pars_tmp(1))/pars_tmp(3)));
  sel_tmp_iq(2)=mfexp(-(square(ages(nages)-pars_tmp(2))/pars_tmp(4)));

  sel_step1=mfexp(-(square(ages-pars_tmp(1))/pars_tmp(3)));
  sel_step2=pars_tmp(5)+(1.0-pars_tmp(5))*(sel_step1-sel_tmp_iq(1))/(1.0-sel_tmp_iq(1));
  sel_step3=mfexp(-(square(ages-pars_tmp(2))/pars_tmp(4)));
  sel_step4=1.0+(pars_tmp(6)-1.0)*(sel_step3-1.0)/(sel_tmp_iq(2)-1.0);
  sel_step5=1.0/ (1.0+mfexp(-(20.0* elem_div((ages-pars_tmp(1)), (1.0+sfabs(ages-pars_tmp(1))))  )));
  sel_step6=1.0/(1.0+mfexp(-(20.0*elem_div((ages-pars_tmp(2)),(1.0+sfabs(ages-pars_tmp(2))))  )));

  Sel_Tmp=elem_prod(sel_step2,(1.0-sel_step5))+
          elem_prod(sel_step5,((1.0-sel_step6)+ elem_prod(sel_step4,sel_step6)) );

  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

//--------------------------------------------------------------------------------
//Spawner-recruit function (Beverton-Holt)
FUNCTION dvariable SR_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const dvariable& SSB)
  //R0=virgin recruitment, h=steepness, spr_F0=spawners per recruit @ F=0, SSB=spawning biomass
  RETURN_ARRAYS_INCREMENT();
  dvariable Recruits_Tmp;
  Recruits_Tmp=((0.8*R0*h*SSB)/(0.2*R0*spr_F0*(1.0-h)+(h-0.2)*SSB));
  RETURN_ARRAYS_DECREMENT();
  return Recruits_Tmp;

//--------------------------------------------------------------------------------
//compute multinomial effective sample size for a single yr
FUNCTION dvariable multinom_eff_N(const dvar_vector& pred_comp, const dvar_vector& obs_comp)
  //pred_comp=vector of predicted comps, obscomp=vector of observed comps
  dvariable EffN_Tmp; dvariable numer; dvariable denom;
  RETURN_ARRAYS_INCREMENT();
  numer=sum( elem_prod(pred_comp,(1.0-pred_comp)) );
  denom=sum( square(obs_comp-pred_comp) );
  if (denom>0.0) {EffN_Tmp=numer/denom;}
  else {EffN_Tmp=-missing;}
  RETURN_ARRAYS_DECREMENT();
  return EffN_Tmp;

//--------------------------------------------------------------------------------
//Likelihood contribution: lognormal
FUNCTION dvariable lk_lognormal(const dvar_vector& pred, const dvar_vector& obs, const dvar_vector& cv, const dvariable& wgt_dat)
  //pred=vector of predicted vals, obs=vector of observed vals, cv=vector of CVs in arithmetic space, wgt_dat=constant scaling of CVs
  //small_number is small value to avoid log(0) during search
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number;
  dvar_vector var(cv.indexmin(),cv.indexmax()); //variance in log space
  var=log(1.0+square(cv/wgt_dat));   // convert cv in arithmetic space to variance in log space
  LkvalTmp=sum(0.5*elem_div(square(log(elem_div((pred+small_number),(obs+small_number)))),var) );
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;

//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
FUNCTION dvariable lk_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const double& ncomp, const double& minSS, const dvariable& wgt_dat)
```

```
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp = number of yrs in matrix, minSS=min N threshold, wgt_dat=scaling of N's
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number=0.00001;
  LkvalTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {if (nsamp(ii)>=minSS)
    {LkvalTmp-=wgt_dat*nsamp(ii)*sum(elem_prod((obs_comp(ii)+small_number),
               log(elem_div((pred_comp(ii)+small_number), (obs_comp(ii)+small_number)))));
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;


//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
FUNCTION dvariable lk_robust_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const double& ncomp, const dvariable& mbin, const double& minSS, const dvariable& wgt
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp = number of yrs in matrix, mbin=number of bins, minSS=min N threshold, wgt_dat=scaling of N's
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number=0.00001;
  LkvalTmp=0.0;
  dvar_matrix Eprime=elem_prod((1.0-obs_comp), obs_comp)+0.1/mbin; //E' of Francis 2011, p.1131
  dvar_vector nsamp_wgt=nsamp*wgt_dat;
  //cout<<nsamp_wgt<<endl;
  for (int ii=1; ii<=ncomp; ii++)
  {if (nsamp(ii)>=minSS)
    {LkvalTmp+= sum(0.5*log(Eprime(ii))-log(small_number+mfexp(elem_div((-square(obs_comp(ii)-pred_comp(ii))) , (Eprime(ii)*2.0/nsamp_wgt(ii)) ))) );
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;


//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//Likelihood contribution: priors
FUNCTION  dvariable neg_log_prior(dvariable pred, const double& prior, dvariable var, int pdf)
  //prior=prior point estimate, var=variance (if negative, treated as CV in arithmetic space), pred=predicted value, pdf=prior type (1=none, 2=lognormal, 3=normal, 4=beta)
    dvariable LkvalTmp;
    dvariable alpha, beta, ab_iq;
    dvariable big_number=1e10;
    LkvalTmp=0.0;
    // compute generic pdf's
    switch(pdf) {
        case 1: //option to turn off prior
          LkvalTmp=0.0;
          break;
        case 2: // lognormal
          if(prior<=0.0) cout << "YIKES: Don't use a lognormal distn for a negative prior" << endl;
          else if(pred<=0) LkvalTmp=big_number=1e10;
          else {
            if(var<0.0) var=log(1.0+var*var) ;      // convert cv to variance on log scale
            LkvalTmp= 0.5*( square(log(pred/prior))/var + log(var) );
          }
        break;
        case 3: // normal
          if(var<0.0 && prior!=0.0) var=square(var*prior);       // convert cv to variance on observation scale
          else if(var<0.0 && prior==0.0) var=-var;               // cv not really appropriate if prior value equals zero
          LkvalTmp= 0.5*( square(pred-prior)/var + log(var) );
          break;
        case 4: // beta
          if(var<0.0) var=square(var*prior);          // convert cv to variance on observation scale
          if(prior<=0.0 || prior>=1.0) cout << "YIKES: Don't use a beta distn for a prior outside (0,1)" << endl;
          ab_iq=prior*(1.0-prior)/var - 1.0; alpha=prior*ab_iq; beta=(1.0-prior)*ab_iq;
          if(pred>=0 && pred<=1) LkvalTmp= (1.0-alpha)*log(pred)+(1.0-beta)*log(1.0-pred)-gammln(alpha+beta)+gammln(alpha)+gammln(beta);
          else LkvalTmp=big_number;
          break;
        default: // no such prior pdf currently available
          cout << "The prior must be either 1(lognormal), 2(normal), or 3(beta)." << endl;
          cout << "Presently it is " << pdf << endl;
          exit(0);
    }
    return LkvalTmp;


//--------------------------------------------------------------------------------
REPORT_SECTION

  if (last_phase())
  {

      // cout<<"start report"<<endl;
      get_weighted_current();
      //cout<<"got weighted"<<endl;
      get_msy();
      //cout<<"got msy"<<endl;
      get_miscellaneous_stuff();
      //cout<<"got misc stuff"<<endl;
      get_per_recruit_stuff();
      //cout<<"got per recruit"<<endl;
      get_effective_sample_sizes();

      cout <<endl;
      cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;
      cout << "BC Fmsy=" << F_msy_out<< "   BC SSBmsy=" << SSB_msy_out <<endl;
      cout <<"F status="<<FdF_msy_end<<endl;
      cout <<"Pop status="<<SdSSB_msy_end<<endl;
      cout << "h="<<steep<<"   RO="<<RO<<endl;
      cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;
     // cout << F_initial << endl;

      //cout << F_initial << sel_mrip << sel_cA << log_F_dev_init_mrip << log_F_dev_init_cA << endl;
```

```
    report << "TotalLikelihood " << fval << endl;
    report << "N" << endl;
    report << N<<endl;
    report << "F" << endl;
    report << F <<endl;

  #include "cobia_make_Robject-baserun.cxx"   // write the S-compatible report

} //endl last phase loop
```

Appendix B Data inputs for the Beaufort Assessment Model

```
##-><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  SEDAR 28 Cobia assessment May 2012
##
##-><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#Starting and ending year of model
1950
2011

#Starting year to estimate recruitment deviation from S-R curve
1975

#Ending year to estimate recruitment deviation from S-R curve
2009

#3 phases of constraints on recruitment deviations:
#allows possible heavier constraint (weights defined later) in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1989
2008

#Ending year for first selectivity period
2011

#Number of ages (12 classes is 1,...,12+) //assumes 12-16 is plus group
12
#Vector of agebins, last is a plus group
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0

#Number length bins used to match length comps and number used to compute plus group
44
16

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
200 230 260 290 320 350 380 410 440 470 500 530 560 590 620 650 680 710 740 770 800 830 860 890 920 950 980 1010 1040 1070 1100 1130 1160 1190 1220 1250 1280 1310 1340 1370 1400 1430 1460 1490
1520 1550 1580 1610 1640 1670 1700 1730 1760 1790 1820 1850 1880 1910 1940 1970

#Max value of F used in spr and msy calculations
3.0
#Number of iterations in spr calculations
10001
#Number of iterations in msy calculations
10001
#Number years at end of time series over which to average sector Fs, for weighted selectivities
3
#Multiplicative bias correction of recruitment (may set to 1.0 for none or negative to compute from recruitment variance)
-1.0
#Number yrs to exclude at end of time series for computing bias correction (end rec devs may have extra constraint)
0

######################Recreational Landings#################################################################################
#Recreational MRFSS CPUE Index (includes shore and private rental modes)
#Starting and ending years of CPUE index
1985
2011
#Observed CPUE and assumed CVs
0.7570   0.5294   0.4279   0.3456   1.4948   0.3675   0.4745   0.5101   0.3620   0.5952   0.7351   0.6884   1.0779   1.1986   1.7307   1.0587   1.4948   0.9490   1.4482   1.4098   1.3906   1.4893   1.0642   1.07
0.6727   0.7188   0.7720   0.5866   0.5792   0.6513   0.6240   0.6588   0.5677   0.5667   0.5635   0.5623   0.5475   0.5339   0.5443   0.5228   0.5213   0.5292   0.5087   0.5095   0.5228   0.4967   0.5161   0.52

#Total recreational landings (includes MRFSS landings+discards, SRHS landings+discards (assume 0.05 discard mortality rate)
#Starting and ending years of landings time series, respectively
1950
2011
#Observed Total Recreational removals (1000s of fish) and assumed CVs (MRFSS + SHRS + dead discards)
2.208    2.376    2.544    2.712    2.880    3.048    3.215    3.383    3.551    3.719    3.887    4.138    4.389    4.640    4.891    5.142    5.354    5.566    5.778    5.990    6.202    6.744    7.285    7.827    8.369    8.910    8.8
0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.0

#Number and vector of years of length compositions for recreational fishery (MRFSS)
31
1981  1982  1983  1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011
#sample size of recreational length comp data by year    (first row observed Ntrips, second row Nfish)
3.0  9.0  5.0  14.0  25.0  28.0  30.0  28.0  41.0  39.0  41.0  42.0  30.0  32.0  43.0  39.0  30.0  37.0  38.0  17.0  33.0  41.0  45.0  42.0  45.0  36.0  62.0  52.0  50.0  69.0  38.0
3.0  9.0  6.0  15.0  30.0  31.0  34.0  29.0  45.0  47.0  42.0  51.0  35.0  40.0  48.0  55.0  37.0  56.0  55.0  22.0  37.0  49.0  50.0  57.0  64.0  41.0  112.0 85.0  61.0  100.0 52.0

#recreational length composition samples (year,lengthbin 3 cm)
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.333333 0.000000 0.000000 0.333333 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.333333 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.133333 0.000000 0.000000 0.000000 0.133333 0.000000 0.133333 0.000000 0.000000 0.000000 0.133333 0.000000 0.000000 0.000000 0.166667 0.000000 0.000000 0.000000 0.000000
0.000000 0.083333 0.000000 0.000000 0.000000 0.000000 0.066667 0.000000 0.066667 0.083333 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.222222
0.111111 0.000000 0.000000 0.000000 0.166667 0.333333 0.000000 0.000000 0.000000 0.166667 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.240000 0.000000 0.000000
0.000000 0.150000 0.040000 0.060000 0.150000 0.000000 0.000000 0.000000 0.100000 0.060000 0.060000 0.000000 0.040000 0.100000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.085185 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.025926 0.000000 0.000000 0.000000 0.024339 0.097354
0.156173 0.051852 0.042593 0.135450 0.000000 0.025926 0.094815 0.000000 0.125750 0.017037 0.025926 0.028395 0.000000 0.034074 0.017037 0.012169 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.029122 0.059737 0.000000 0.000000 0.119474 0.000000 0.000000 0.000000 0.000000 0.000000 0.050179 0.043011
0.021505 0.138441 0.017921 0.022401 0.067204 0.089606 0.032258 0.017921 0.022401 0.089606 0.089606 0.000000 0.000000 0.000000 0.089606 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.066176 0.016544 0.000000 0.000000 0.000000 0.066176 0.000000 0.000000 0.000000 0.000000 0.000000 0.082721 0.000000 0.000000 0.000000 0.033088 0.066176 0.000000 0.000000 0.000000 0.012868 0.033088
0.066176 0.117647 0.045956 0.000000 0.091912 0.000000 0.036765 0.093750 0.034926 0.016544 0.007353 0.023897 0.022059 0.066176 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.027094 0.000000 0.027094 0.000000 0.000000 0.000000 0.000000 0.016092 0.000000 0.000000 0.000000 0.027094 0.107553 0.027094 0.151560
0.124466 0.000000 0.108374 0.080460 0.000000 0.000000 0.043186 0.027094 0.000000 0.043186 0.027094 0.000000 0.108374 0.054187 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.076768 0.076768 0.000000 0.000000 0.000000 0.000000 0.000000 0.025589 0.000000 0.000000 0.000000 0.038384 0.000000 0.000000 0.009596 0.022391 0.051178 0.000000 0.000000
0.186173 0.066229 0.115196 0.063973 0.024949 0.076768 0.024961 0.038721 0.000000 0.000000 0.000000 0.000000 0.012795 0.000000 0.025589 0.000000 0.000000 0.025589 0.038384 0.000000 0.000000 0.000000
```

```
0.000000 0.000000 0.000000 0.011550 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.027720 0.000000 0.043121 0.038501 0.000000 0.000000 0.080851 0.023100 0.019504 0.000000 0.037082 0.080851
0.121337 0.050922 0.085532 0.077305 0.036383 0.087589 0.065461 0.063333 0.006738 0.000000 0.000000 0.026950 0.000000 0.000000 0.000000 0.000000 0.016170 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.046459 0.000000 0.000000 0.000000 0.046459 0.000000 0.009292 0.092919 0.000000 0.000000 0.000000 0.000000 0.000000 0.032842 0.000000 0.000000 0.127363 0.000000 0.072186
0.000000 0.076832 0.125174 0.046459 0.051426 0.013017 0.062894 0.023230 0.108405 0.000000 0.000000 0.009292 0.000000 0.009292 0.000000 0.000000 0.000000 0.046459 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.008374 0.000000 0.000000 0.008374 0.000000 0.000000 0.000000 0.000000 0.016748 0.120098
0.079248 0.109010 0.097193 0.009570 0.059816 0.013399 0.124496 0.125054 0.017945 0.033088 0.043067 0.008374 0.084273 0.008374 0.033497 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.009977 0.000000 0.046429
0.050397 0.137755 0.046429 0.131746 0.034921 0.069841 0.100794 0.151190 0.089796 0.075850 0.009977 0.000000 0.000000 0.009977 0.000000 0.034921 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.010764 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.012500 0.012500
0.000000 0.027986 0.050000 0.152369 0.152083 0.012500 0.026790 0.114097 0.125401 0.163079 0.020332 0.062191 0.009568 0.009568 0.000000 0.000000 0.028704 0.000000 0.000000 0.000000 0.009568 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.022619 0.037500
0.030258 0.067460 0.149479 0.028646 0.121288 0.074818 0.036624 0.035012 0.085243 0.104167 0.035012 0.000000 0.006366 0.006366 0.028646 0.044560 0.000000 0.085938 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.007959 0.007959 0.000000 0.015917 0.003979 0.003979 0.015917 0.000000 0.000000 0.000000 0.000000 0.003979 0.000000 0.087546 0.038917
0.131318 0.098489 0.242007 0.000000 0.003979 0.080036 0.000000 0.097281 0.020679 0.000000 0.040256 0.009736 0.029182 0.003979 0.021886 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.035018 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.048649 0.000000
0.000000 0.076507 0.125017 0.035620 0.116424 0.061816 0.019404 0.086071 0.019404 0.152044 0.087318 0.040748 0.087318 0.029106 0.000000 0.014553 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.041964 0.062054 0.013393
0.020089 0.041964 0.048065 0.005595 0.039966 0.199936 0.025383 0.053954 0.116008 0.049664 0.030179 0.016786 0.156667 0.011190 0.000000 0.011190 0.000000 0.027976 0.027976 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.016234 0.048701 0.000000 0.000000 0.146104 0.016234 0.016234 0.032468 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.010101 0.233045 0.000000 0.032468 0.042569 0.173160 0.016234 0.000000 0.000000 0.000000 0.129870 0.000000 0.000000 0.000000 0.043290 0.000000 0.043290 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.079545 0.000000 0.000000 0.000000 0.000000 0.000000 0.045455
0.045455 0.000000 0.045455 0.000000 0.000000 0.053030 0.045455 0.000000 0.000000 0.098485 0.204545 0.000000 0.000000 0.125000 0.000000 0.079545 0.045455 0.000000 0.000000 0.000000 0.053030 0.079545 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.027027 0.000000 0.052365
0.000000 0.038664 0.034910 0.087275 0.049456 0.118056 0.023273 0.157095 0.170420 0.052365 0.011637 0.000000 0.000000 0.034910 0.000000 0.000000 0.011637 0.026182 0.000000 0.000000 0.000000 0.104730
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.040816
0.119898 0.040816 0.067035 0.054847 0.012188 0.032596 0.000000 0.000000 0.012188 0.012188 0.046769 0.176729 0.109694 0.000000 0.000000 0.146259 0.054847 0.000000 0.036565 0.000000 0.000000 0.036565
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.017091
0.009495 0.048485 0.072162 0.314283 0.099758 0.055071 0.000000 0.085455 0.085455 0.052222 0.020000 0.085455 0.026586 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.028485 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.038743 0.000000
0.116228 0.000000 0.022189 0.116228 0.033812 0.010566 0.151874 0.084323 0.135599 0.000000 0.069810 0.023392 0.000000 0.061988 0.022189 0.101435 0.000000 0.000000 0.000000 0.000000 0.000000 0.011623
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.109890 0.000000 0.000000 0.000000 0.006868 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.054945
0.000000 0.017857 0.034341 0.108516 0.064103 0.039835 0.116758 0.036630 0.000000 0.018315 0.054945 0.076923 0.051282 0.021978 0.018315 0.168498 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.016260 0.016260 0.271022 0.000000 0.197677 0.032230 0.075203 0.100987 0.000000 0.032230 0.000000 0.064750 0.000000 0.128920 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.064460 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.094637 0.055205 0.056782 0.123554 0.147213 0.023659 0.104101 0.000000 0.019979 0.019979 0.066246 0.094637 0.056782 0.023659 0.000000 0.000000 0.000000 0.000000 0.094637 0.018927 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.080392 0.000000
0.091877 0.008824 0.000000 0.000000 0.181092 0.000000 0.109524 0.000000 0.183754 0.080392 0.000000 0.011485 0.011485 0.080392 0.000000 0.000000 0.080392 0.000000 0.080392 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.040301 0.000000
0.094035 0.161202 0.076844 0.107468 0.080601 0.060451 0.134335 0.040301 0.060451 0.060451 0.000000 0.000000 0.056694 0.000000 0.000000 0.026867 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.018419 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.055257 0.000000 0.006907
0.055257 0.030128 0.013716 0.097286 0.050724 0.236558 0.068888 0.079815 0.064612 0.050983 0.011402 0.010817 0.015729 0.048094 0.000000 0.020465 0.018419 0.039798 0.000000 0.006724 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.111607 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.029762 0.065104 0.007440
0.027902 0.014881 0.027530 0.126488 0.119048 0.014881 0.131696 0.184152 0.029762 0.027902 0.000000 0.000000 0.000000 0.037202 0.000000 0.007440 0.007440 0.000000 0.000000 0.000000 0.007440 0.022321

#Number and vector of years of age compositions for recreational fishery--(no age comps for 1998)
27
1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011
###sample sizes of age comps by year (first row observed Ntrips, second row Nfish); assume ntrips same as nfish (1 fish per trip) and then cap at a sample size of 200
3.0  2.0  22.0  18.0  17.0  78.0  101.0  16.0  20.0  16.0  16.0  10.0  31.0  20.0  130.0  111.0  72.0  27.0  7.0  7.0  125.0  81.0  200.0  200.0  200.0  200.0  200.0
3.0  2.0  22.0  18.0  17.0  78.0  101.0  16.0  20.0  16.0  16.0  10.0  31.0  20.0  130.0  111.0  72.0  27.0  7.0  7.0  125.0  81.0  397.0  327.0  311.0  330.0  307.0

#age composition samples (year,age) from recreational fishery--combined across gears--last age is a plus group (1 to 12+)
0.3333 0.0000 0.6667 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.5000 0.0000 0.0000 0.0000 0.0000 0.5000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.1364 0.3182 0.0455 0.2727 0.1818 0.0000 0.0000 0.0455 0.0000 0.0000 0.0000
0.1111 0.5000 0.0000 0.1111 0.0000 0.1667 0.1111 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.2941 0.0588 0.1176 0.2353 0.0588 0.0588 0.1176 0.0588 0.0000 0.0000 0.0000
0.0000 0.1282 0.2436 0.2308 0.0641 0.0897 0.0641 0.0769 0.0769 0.0000 0.0128 0.0128
0.0000 0.0000 0.4854 0.1650 0.1165 0.0388 0.0388 0.0388 0.0291 0.0583 0.0097 0.0194
0.0000 0.0000 0.0625 0.3750 0.1875 0.1250 0.0000 0.0625 0.0000 0.1250 0.0625 0.0000
0.0000 0.1500 0.2000 0.1000 0.1500 0.2000 0.2000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.3125 0.0625 0.0000 0.2500 0.0000 0.1250 0.0625 0.0625 0.0000 0.1250
0.0000 0.0000 0.0625 0.3750 0.1250 0.0625 0.1250 0.0625 0.0000 0.0000 0.0625 0.1250
0.0000 0.0000 0.0000 0.0000 0.2000 0.1000 0.0000 0.4000 0.2000 0.0000 0.1000 0.0000
0.0000 0.0000 0.1290 0.0323 0.1290 0.2903 0.1935 0.0968 0.0968 0.0000 0.0000 0.0323
0.0000 0.2500 0.2000 0.1000 0.1000 0.1000 0.0500 0.0000 0.1000 0.0000 0.1000 0.0000
0.0000 0.0077 0.0385 0.3154 0.0538 0.2615 0.0154 0.0231 0.1385 0.0615 0.0154 0.0692
0.0000 0.0090 0.0721 0.0360 0.2252 0.0360 0.2793 0.0270 0.0090 0.1171 0.1081 0.0811
0.0000 0.0000 0.3750 0.1528 0.0000 0.2083 0.0278 0.1250 0.0000 0.0139 0.0694 0.0278
0.0000 0.0000 0.1111 0.2963 0.0741 0.0741 0.3333 0.0370 0.0370 0.0000 0.0000 0.0370
0.0000 0.0000 0.0000 0.1429 0.2857 0.1429 0.0000 0.4286 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.4286 0.5714 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0320 0.2560 0.1440 0.1440 0.1760 0.1840 0.0000 0.0480 0.0160 0.0000 0.0000
0.0000 0.0864 0.0864 0.3210 0.1235 0.1481 0.1111 0.0494 0.0123 0.0123 0.0123 0.0370
0.0025 0.0101 0.7305 0.0252 0.1083 0.0277 0.0353 0.0151 0.0202 0.0076 0.0126 0.0050
0.0000 0.0061 0.1529 0.6086 0.0214 0.0673 0.0398 0.0428 0.0275 0.0275 0.0031 0.0031
0.0000 0.0129 0.1190 0.2733 0.3633 0.0193 0.1190 0.0193 0.0193 0.0225 0.0225 0.0096
0.0000 0.0121 0.2455 0.1818 0.2121 0.2212 0.0182 0.0667 0.0303 0.0030 0.0000 0.0091
0.0000 0.0130 0.1792 0.2476 0.1596 0.1466 0.1466 0.0130 0.0423 0.0130 0.0098 0.0293

#####################################Commercial (all sectors)#############################################################
#Starting and ending years for landings time series
1950
2011
#commercial landings vector (1000 lb whole weight) and assumed CVs (landings + discards)
11.400   11.800   3.800   13.700   28.200   9.200   27.100   48.600   25.500   48.900   30.700   38.700   41.100   49.900   24.500   19.900   12.100   12.800   10.900   9.000   9.200   14.400   7.000   4.600   5
0.05     0.05    0.05     0.05     0.05     0.05    0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05     0.05    0.05    0.05     0.05     0.05    0.05    0
#Number and vector of years of commercial length compositions (pooled over gears; pooled over years and weighted by number trips)
1
2011  #Used to represented pooled years from 1982 to 2011
#Sample size of length comp data (first row avg number observed trips; second row total  number observed fish
9.0   #average number of trips (1982 to 2011)
438.0  #total number of fish (1982 to 2011)
#commercial length comps (pooled over years, 3 cm length bins)
0.031304   0.012965   0.024177   0.028854   0.025825   0.014557   0.013999   0.000000   0.022824   0.005053   0.008267   0.024587   0.022654   0.044792   0.011057   0.001621   0.011727   0.009026   0.008167   0.

#Number and vector of years of length compositions for commercial fishery; used to weight annual predictions before pooling
26
1982  1984  1985  1987  1988  1989  1990  1991  1992  1993  1994  1995  1997  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010 2011
###sample sizes of comm length comps by year for pooling (this is number of n.trips)
2.0  12.0  9.0  5.0  14.0  4.0  3.0  10.0  1.0  5.0  6.0  3.0  3.0  5.0  4.0  7.0  3.0  6.0  17.0  21.0  13.0  12.0  33.0  18.0  7.0  14.0

#number and vector of years of age comp data for comm fishery (pooled)
```

```
1
2011  #used to represent pooled years from 1986 to 2011
#Sample size of age comp data (first row avg number observed trips; second row average number observed fish)
7.0     #average number of fish (1986-2011); assumes 1 fish per trip
120.0  #total number of fish (1986-2011)
#commercial age comps (pooled over years, ages 1 to 12+)
0.042735   0.128205   0.222222   0.230769   0.076923   0.085470   0.051282   0.042735   0.042735   0.008547   0.000000   0.068376

###Number and vector of years of age compositions for commercial fishery; used to weight annual predictions before pooling
18
1986  1989  1990  1991  1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011
###sample sizes of comm age comps by year for pooling (this is number of fish since no n.trips)
1.0  4.0  3.0  1.0  5.0  9.0  7.0  7.0  36.0  2.0  2.0  6.0  2.0  11.0  5.0  3.0  5.0  11.0

########################################Headboat CPUE index#################################################################
#Starting and ending years of HB index
1981
2011
#Observed CPUE and assumed CVs
0.72   0.71   0.81   0.36   0.36   0.71   1.18   0.88   0.81   0.55   1.72   1.34   1.05   1.19   1.32   0.56   0.94   0.86   0.90  1.28   1.34   0.90  1.11   1.08   1.08   0.94   1.54   1.96   0.93   0.88   0.9
0.25   0.26   0.25   0.31   0.56   0.27   0.19   0.21   0.25   0.26   0.17   0.16   0.15   0.15   0.14   0.20   0.17   0.15   0.18  0.17   0.17   0.16  0.19   0.16   0.19   0.2    0.14   0.15   0.21
0.17   0.22

########################################SC Commercial logbook CPUE index#####################################################
1998
2011
#Observed CPUE and assumed CVs
1.16   1.39   1.04   1.21   0.97   0.73   1.20  0.96   0.95   1.11   0.79   1.05   0.73   0.73
0.32   0.31   0.24   0.25   0.33   0.22   0.25  0.23   0.21   0.24   0.23   0.23   0.27   0.23

#####################Biological input ###########################################################################
#VonBert params (Linf, K, t0), units in mm TL for all fish--all fish, inverse weighted by n, diaz-corrected
1324.4
0.27
-0.47
#Standard errors of vonBert param (Linf, K, t0), applied if params are estimated
115.7
0.073
0.192
#CV of length at age
0.131
#standard error of CV of length at age, applied if CV is estimated
0.0308

#length-weight (FL-whole wgt) coefficients a and b, W=aL^b, (W in kg, FL in mm)--sexes combined
2.00E-9
3.28
#weight-gonad weight (whole wgt-gondad weight) coefficients a and b, GW=a+b*W (units=g) not used in this model
-9.1
1.7

#time-invariant vector of % maturity-at-age for females (ages 1-12+)
0.0  0.5  0.75  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00

#time-invariant vector of proportion female (ages 1-12+)--assume 50:50 sex ratio
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
#time of year (as fraction) for spawning: end of May  (peaks SC 5/11, peaks NC 6/10)
0.42
#age-dependent natural mortality at age (ages 1-12+)
0.559   0.418  0.350  0.312  0.288  0.271  0.261  0.253  0.247  0.242  0.239  0.238

#age-independent natural mortality (used only to compute MSST=(1-M)SSBmsy)
0.26
#SE of age-independent natural mortality (leftover from tilefish)
0.024
#Max observed age
16
#Spawner-recruit parameters
#steepness (fixed or initial guess) (0.75 from meta-analysis)
0.75  #mu=alpha/(alpha+beta);  alpha and beta from beta distn
#SE of steepness (from meta-analysis)
0.14  #var=(alpha*beta)/[(alpha+beta)^2*(alpha+beta+1)];  alpha and beta from beta distn
#log_R0 - log virgin recruitment
12.0
# R autocorrelation
0.0
# SD of recruitment in log space
0.6
# SE of SD recruitment
0.15

##################Parameter values and initial guesses#########################################################################
###Selectivity parameters.
###Initial guess must be within boundaries.
# Initial guesses initialized near solutions from catch curve analysis
# zero in slope2 provides logistic selectivity

#recreational selectivity starting values (from catch curve analysis)
2.75  # age at 50% selectivity
4.08  # slope of ascending limb

#commercial selectivity starting values (from catch curve analysis--data pooled 1986-2011)
2.65  # age at 50% selectivity
1.96  # slope of ascending limb

###################Likelihood Component Weighting###############################################################################
##Weights in objective fcn  (commented wgts are those after correcting M vector--did not use)
1.0               #landings
0.117                #mrip length comps
0.117                #commercial length comps
0.198           #mrip age comps
0.198                #commercial age comps
1.0               #mrip index
0.494                #HB index
```

```
0.954                #SC charterboat index
1.0     #S-R residuals
0.0     #constraint on early recruitment deviations
0.0     #constraint on ending recruitment deviations
0.0         #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization) fULL F summed over fisheries
0.0     #weight on tuning F (penalty not applied in final phase of optimization)

#log catchabilities (initial guesses)
-10.0      #MRFSS CPUE
-10.0      #HB CPUE
-10.0      #SC logbook CPUE
#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
-1
##annual positive rate of increase on all fishery dependent q's due to technology creep
0.0
# DD q switch: Integer value (choose estimation phase, negative value turns it off)
-1
##density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)
0.0
##SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)
0.128
#Age to begin counting D-D q (should be age near full exploitation)
2
#Random walk switch:Integer value (choose estimation phase, negative value turns it off)
-3
#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence
0.03
0.03
0.03

##log mean F's (initial guesses)
-4.0 #commercial longline (leftover from tilefish
-4.0                    #commercial
#Initial F (Input here, could also use mean of first few years)
0.005
#Tuning F (not applied in last phase of optimization)
0.2
#Year for tuning F
2010

##threshold sample sizes for length comps (set to 99999.0 if sel is fixed)
1.0 #MRFSS
1.0 #commercial

#threshold sample sizes (greater than or equal to) for age comps
1.0 #MRFSS
1.0 #commercial

#Ageing error matrix (columns are true age 1-12+, rows are ages as read for age comps: columns should sum to one)
0.982 0.018 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.018 0.964 0.018 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.018 0.964 0.018 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.018 0.964 0.018 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.018 0.964 0.018 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.018 0.964 0.018 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.018 0.964 0.018 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.018 0.964 0.018 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.018 0.964 0.018 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.018 0.964 0.018 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.018 0.964 0.018
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.018   0.982
999 #end of data file flag
```