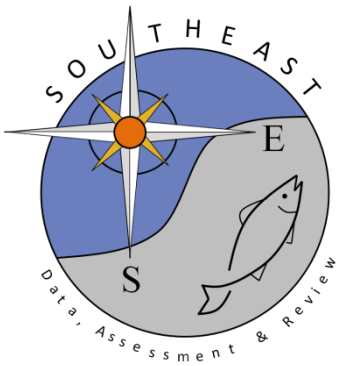
The Beaufort Assessment Model (BAM) with application to blueline tilefish: mathematical description, implementation details, and computer code

Sustainable Fisheries Branch, National Marine Fisheries Service,
Southeast Fisheries Science Center – Beaufort Lab (contact: Kevin Craig)

SEDAR32-RW-01

Submitted: 12 August 2013



Please cite this document as:

Sustainable Fisheries Branch, National Marine Fisheries Service, Southeast Fisheries Science Center – Beaufort Lab. 2013. The Beaufort Assessment Model (BAM) with application to blueline tilefish: mathematical description, implementation details, and computer code. SEDAR32-RW01. SEDAR, North Charleston, SC. 42 pp.

The Beaufort Assessment Model (BAM) with application to blueline tilefish: 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, golden tilefish, and cobia.

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 $1 - 15^+$, where the oldest age class 15^+ allowed for the accumulation of fish (i.e., plus group).

Initialization Initial (1974) abundance at age was computed in the model assuming an equilibrium age structure and initial fishing mortality rate. The equilibrium age structure was computed for ages $1 - 15^+$ based on natural and fishing mortality (F), where initial F was estimated by the model. This was based on the assumption by the AW panel that the stock was lightly exploited (but less than virgin) prior to the 1970s.

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 Charnov et al. (2013). The Charnov et al. (2013) approach inversely relates the natural mortality at age to mean weight at age W_a by the power function $\mathrm{M}_a{=}\alpha W_a^{\beta}$, where α is a scale parameter and β is a shape parameter. The Charnov et al. (2013) approach is an update to an earlier approach based on point estimates of α and β for oceanic fishes described in Lorenzen (1996). As in previous SEDAR assessments, the Charnov estimates of M_a were rescaled to provide the same fraction of fish surviving from age-1 through the oldest observed age (43 yr) as would occur with constant $M = 0.10$ 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 $L_\infty = 600.3$ mm, $k = 0.33$, and $t_0 = -0.50$ yr. To convert age of fish landed in the fishery to mean size, a power function, $\mathrm{FL}_a{=}\alpha Age_a^{\beta}$, was used where α is a scale parameter and β is a shape parameter. A single power function was used to match landings in the commercial handline and longline fisheries because length at age was similar between these two fisheries, and a second power function was used for the recreational fishery which landed smaller, younger fish.

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 0.1, 0.25, and 0.5 respectively.

Spawning stock Spawning stock was modeled using total mature female biomass measured at the time of peak spawning. For blueline tilefish, peak spawning was considered to occur at the end of 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 1974–2009 only. These deviations were constrained to sum to 1.0. 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 three time series of combined landings plus discards from 1974-2011: the general recreational fleet (pooled over gears), the commerical handline fleet, and the commercial longline fleet. Discards were a small proportion of landings and were combined with landings from the respective fleet. The discard mortality rate was assumed to be 1.0 as recommended by the DW.

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 (1974).

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 all fleets were modeled as flat-topped, using a two parameter logistic function. Selectivities of the handline and longline indices were assumed the same as the respective fisheries. Selectivity of the recreational headboat index was assumed the same as that of the general recreational fleet because all use hook and line gear.

Weak priors were used for estimating selectivity functions. These priors assumed normal distributions with CV = 0.5 (handline and longline fisheries) or CV=0.25 (general recreational fishery) and were intended to provide limited information to help the optimization routine during model execution. 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 optimum value. Diffuse priors help avoid this situation.

Indices of abundance The model was fit to three indices of relative abundance: the headboat index (1980-1992), the commercial longline index (1993-2004), and the commercial handline index (1993-2010). Predicted indices were conditional on selectivities 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. 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 blueline tilefish, 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 and age 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 age compositions from the handline and longline fisheries were assumed equal to the annual number of trips sampled. 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 composition was available for the recreational fishery, this approach could not be used to derive a weight for this data sources. Therefore, the weight on the recreational age composition was assumed to be the same as that for the commercial handline fishery because both use similar gear and fish in a comparable manner.

In addition, the compound objective function included several penalties or prior distributions applied to the initial fishing mortality rate, the 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 The general BAM model structure has been through multiple SEDAR reviews. As an additional measure of quality control, blueline tilefish 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 blueline tilefish 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.

Charnov, E. L., H. Gislason, and J. Pope. 2013. Evolutionary assembly rules for fish life histories. Fish and Fisheries **14**:213–224.

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. 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.

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 blueline tilefish. 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	$y \in \{1974 \dots 2011\}$
Index of ages	a	$a \in \{1, 2 \dots A\}$, where $A = 15^{+}$
Index of length bins	l	$l \in \{1, 2 \dots 16\}$
Length bins	l'	$l' \in \{260, 290, \dots, 920\text{mm}\}$, with midpoint of 30mm bin used to match length compositions. Largest 7 length bins ($FL \geq 710$ mm) treated as a plus group, but retained for weight calculations.
Index of fisheries	f	$f \in \{1, 2, 3\}$ where 1 = general recreational, 2 = commercial handline, 3 = commercial longline
Index of CPUE	u	$u \in \{1, 2, 3\}$ where 1 = Headboat, 2 = handline, 3 = longline
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), $y \in \{1980 \dots 1992\}$ $u = 2$, Commercial handline (whole weight), $y \in \{1993 \dots 2010\}$ $u = 3$, Commercial longline (whole weight), $y \in \{1993 \dots 2004\}$ Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean.
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1, 2, 3\}$ 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 Charnov et al. (2013). Charnov M_a then rescaled based on Hoenig estimate.
Population Model		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Proportion females mature at age	m_a	Increasing with age {0.1, 0.25, 0.5, 1 ...,1} for ages $1-15^{+}$; assumed constant across years.
Spawning date	t_{spawn}	Fraction denoting the proportional time of year when spawning occurs. Set to 0.42 for blueline tilefish by assuming peak spawning occurs in the end of May.
Mean length at age of population	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.
Mean length at age of landings	$l^L_{(f),a}$	Total length (midyear); $l^L_{(f),a} = \theta_1 (a^L_{(f)})^{\theta_2}$ where θ_1 and θ_2 are parameters
CV of $l^L_{(f),a}$	$\widehat{c}^{\lambda}_{(f),a}$	Estimated coefficient of variation of growth, assumed constant across ages
SD of $l^L_{(f),a}$	$\sigma^{\lambda}_{(f),a}$	Standard deviation of growth, assumed constant across ages.
Age–length conversion of landings	$\psi^L_{(f),a,l}$	$\psi^L_{(f),a,l} = \frac{1}{\sqrt{2\pi}(\sigma^{\lambda}_{(f),a})} \frac{\exp\left[-\left(l'_{(f),l} - l_{(f),a}\right)^2\right]}{\left(2\left(\sigma^{\lambda}_{(f),a}\right)^2\right)}$, the Gaussian density function. Matrix $\psi^L_{(f)}$ is rescaled to sum to one within ages, with the largest size a plus group. This matrix is constant across years.
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 $w^L_{(f),a,y} = \theta_1 (l^L_{(f),a})^{\theta_2}$
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 headboat index was 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	$N_{a,y}$	$N_{1,1974} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}\phi_{init}-0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)\phi_{init}}$ $\widehat{N}_{1+,1974}$ equilibrium conditions expected given assumptions about initial fishing mortality (described below) $N_{0,y+1} = \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\text{for} \quad y \geq 1974$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (0 \dots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ $\widehat{R}_0$ (asymptotic maximum recruitment) is an estimated parameter of the spawner-recruit curve, and $\widehat{R}_y$ are estimated annual recruitment deviations in log space. 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-32 baserun, $h = 0.84$ 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 estimated by the model and s_a^{init} is the F-weighted average of recreational and commercial selectivity for the first assessment year.
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	$L'_{f,a,y}$	$L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}}N_{a,y}[1-\exp(-Z_{a,y})]$
Landings at age in whole weight	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y}L'_{f,a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Index catchability	$q_{u,y}$	$q_{u,1981} = \widehat{q}_u^0 f(\text{density})$ $q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ 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-32 blueline tilefish assessment. Density dependence and random walks were not applied in the baserun.
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \begin{cases} \sum_a L'_{f,a,y} & :f = 1, \\ \sum_a L''_{f,a,y} & :f = 2, \\ \sum_a L'''_{f,a,y} & :f = 3 \end{cases}$
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	$\Lambda_5 = \omega_5\left[\frac{[R_{1974}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y>1974}^{2009}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\widehat{\sigma}_R)\right]$ 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-32 base run).
Additional constraint on early recruitment deviations	Λ_6	$\Lambda_6 = \omega_6\left[\frac{[R_{1974}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y=1975}^{Y_1}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\widehat{\sigma}_R)\right]$ 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-32 blueline tilefish 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-32 blueline tilefish 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-32 blueline tilefish baserun, thus ω_8=0.0.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Penalty on initial age structure	Λ_9	$\Lambda_9 = \sum_{a=1}^{A} (\widehat{N}_{a,1974} - N_a^{eq})^2$ where N_a^{eq} is the equilibrium age structure given the initial F, as defined previously. ω_7=0.0 for the SEDAR-32 blueline tilefish 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-32 blueline tilefish 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 32 Blueline Tilefish assessment May 2013
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
DATA_SECTION

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

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: set-up section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

// Starting and ending year of the model (year data starts)
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_number lenbins_width;  //width of length bins (mm)

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

//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;

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: observed data section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

//###################Recreational fishery #########################

// Landings  (1000 of fish)
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);

init_int nyr_mrip_agec_pool;   //no. yrs to pool predicted mrip age comps to match pooled observations
init_ivector yrs_mrip_agec_pool(1,nyr_mrip_agec_pool);   //specific yrs to pool age comps over
init_vector nsamp_mrip_agec_pool(1,nyr_mrip_agec_pool); //sample sizes by year for pooling
```

```
//################Commercial handline fishery#######################################
//Landings (1000 lbs whole weight)
 init_int styr_cHL_L;
 init_int endyr_cHL_L;
 init_vector obs_cHL_L(styr_cHL_L,endyr_cHL_L);
 init_vector cHL_L_cv(styr_cHL_L,endyr_cHL_L);

//Length compositions (3 cm bins) of landings
 init_int nyr_cHL_lenc;
 init_ivector yrs_cHL_lenc(1,nyr_cHL_lenc);
 init_vector nsamp_cHL_lenc(1,nyr_cHL_lenc);
 init_vector nfish_cHL_lenc(1,nyr_cHL_lenc);
 init_matrix obs_cHL_lenc(1,nyr_cHL_lenc,1,nlenbins);

//Age compositions (1 to 15+)
 init_int nyr_cHL_agec;
 init_ivector yrs_cHL_agec(1,nyr_cHL_agec);
 init_vector nsamp_cHL_agec(1,nyr_cHL_agec);
 init_vector nfish_cHL_agec(1,nyr_cHL_agec);
 init_matrix obs_cHL_agec(1,nyr_cHL_agec,1,nages);

//################Commercial longline fishery#######################################
//Landings (1000 lbs whole weight)
 init_int styr_cLL_L;
 init_int endyr_cLL_L;
 init_vector obs_cLL_L(styr_cLL_L,endyr_cLL_L);
 init_vector cLL_L_cv(styr_cLL_L,endyr_cLL_L);

//Length compositions (3 cm bins) of landings
 init_int nyr_cLL_lenc;
 init_ivector yrs_cLL_lenc(1,nyr_cLL_lenc);
 init_vector nsamp_cLL_lenc(1,nyr_cLL_lenc);
 init_vector nfish_cLL_lenc(1,nyr_cLL_lenc);
 init_matrix obs_cLL_lenc(1,nyr_cLL_lenc,1,nlenbins);

//Age compositions (1 to 15+)
 init_int nyr_cLL_agec;
 init_ivector yrs_cLL_agec(1,nyr_cLL_agec);
 init_vector nsamp_cLL_agec(1,nyr_cLL_agec);
 init_vector nfish_cLL_agec(1,nyr_cLL_agec);
 init_matrix obs_cLL_agec(1,nyr_cLL_agec,1,nages);

//####################### 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);

//####################### Com Handline CPUE data ##############################
//CPUE
 init_int styr_cHL_cpue;
 init_int endyr_cHL_cpue;
 init_vector obs_cHL_cpue(styr_cHL_cpue,endyr_cHL_cpue);
 init_vector cHL_cpue_cv(styr_cHL_cpue,endyr_cHL_cpue);

//####################### Com Longline CPUE data ##############################
//CPUE
init_int styr_cLL_cpue;
init_int endyr_cLL_cpue;
init_vector obs_cLL_cpue(styr_cLL_cpue,endyr_cLL_cpue);//Observed CPUE
init_vector cLL_cpue_cv(styr_cLL_cpue,endyr_cLL_cpue); //CV of cpue

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: parameter section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

//#################Single Parameter values and initial guesses ################################
// Von Bert parameters in TL mm all fish
init_vector set_Linf(1,7);
init_vector set_K(1,7);
init_vector set_t0(1,7);
//CV of length at age and its standard error all fish
init_vector set_len_cv(1,7);

//these assume a power function to fit the fishery growth curve (separate for HL-LL and mrip)
init_vector set_agepar_a_F(1,7);          //HL-LL power fct intercept
init_vector set_agepar_b_F(1,7);          //HL-LL power fct slope
init_vector set_len_cv_F(1,7);         //HL-LL power fct CV

init_vector set_agepar_a_mrip(1,7);        //mrip power fct intercept
init_vector set_agepar_b_mrip(1,7);        //mrip power fct slope
init_vector set_len_cv_mrip(1,7);      //mrip power fct CV

//age-independent: used only for MSST and to scale age dependent M, prior if M is estimated
init_vector set_M_constant(1,7);
//Spawner-recruit parameters (Initial guesses or fixed values)
init_vector set_steep(1,7);         //recruitment steepness
init_vector set_log_R0(1,7);        //recruitment R0
init_vector set_R_autocorr(1,7);    //recruitment autocorrelation
init_vector set_rec_sigma(1,7);     //recruitment standard deviation in log space

//Initial guesses or fixed values of estimated selectivity parameters
//recreational (all)
init_vector set_selpar_L50_mrip(1,7);
init_vector set_selpar_slope_mrip(1,7);
//commercial HL
init_vector set_selpar_L50_cHL(1,7);
init_vector set_selpar_slope_cHL(1,7);
//commercial LL
init_vector set_selpar_L50_cLL(1,7);
init_vector set_selpar_slope_cLL(1,7);
//////--index catchability----------------------------------------------------------------------------
init_vector set_log_q_cLL(1,7);      //catchability coefficient (log) for comm LL cpue index
```

```
init_vector set_log_q_hb(1,7);        //catchability coefficient (log) for headboat cpue index
init_vector set_log_q_cHL(1,7);         //catchability coefficient (log) for comm HL cpue index
//initial F
init_vector set_F_init(1,7);        //initial F, scaled by F_init_ratio

////--F's--------------------------------
init_vector set_log_avg_F_mrip(1,7);
init_vector set_log_avg_F_cHL(1,7);
init_vector set_log_avg_F_cLL(1,7);
//##################Dev Vector Parameter values (vals) and bounds #################################
//--F vectors---------------------------
init_vector set_log_F_dev_mrip(1,3);
init_vector set_log_F_dev_cHL(1,3);
init_vector set_log_F_dev_cLL(1,3);
init_vector set_log_rec_dev(1,3);

init_vector set_log_Nage_dev(1,3);

init_vector set_log_F_dev_mrip_vals(styr_mrip_L,endyr_mrip_L);
init_vector set_log_F_dev_cHL_vals(styr_cHL_L,endyr_cHL_L);
init_vector set_log_F_dev_cLL_vals(styr_cLL_L,endyr_cLL_L);
init_vector set_log_rec_dev_vals(styr_rec_dev,endyr_rec_dev);
init_vector set_log_Nage_dev_vals(2,nages);

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: likelihood weights section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

init_number set_w_L;
init_number set_w_lc_mrip;
init_number set_w_lc_cHL;
init_number set_w_lc_cLL;
init_number set_w_ac_mrip;
init_number set_w_ac_cHL;
init_number set_w_ac_cLL;
init_number set_w_I_cLL;
init_number set_w_I_hb;            //weight for headboat index
init_number set_w_I_cHL;            //weight for comm HL 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)

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: miscellaneous stuff section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

//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
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 active_f_obs(1,nages);             //proportion active spawners

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 max_obs_age;        //max observed age, used to scale M

//Spawner-recruit parameters (Initial guesses or fixed values)
init_number SR_switch;

//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 indepenent
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_cLL_var;    //assumed variance of RW q
init_number set_q_RW_hb_var;    //assumed variance of RW q
init_number set_q_RW_cHL_var;    //assumed variance of RW q

//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_cHL_lenc;
init_number minSS_cLL_lenc;

//threshold sample sizes for age comps
init_number minSS_mrip_agec;
init_number minSS_cHL_agec;
init_number minSS_cLL_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 use_landings_growth;

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);
   }
   else
   {
    cout << "Data File read correctly" << endl;
   }
 END_CALCS


PARAMETER_SECTION

 LOCAL_CALCS
  //POPULATION
  const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
  const double K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
  const double t0_LO=set_t0(2); const double t0_HI=set_t0(3); const double t0_PH=set_t0(4);
  const double len_cv_LO=set_len_cv(2); const double len_cv_HI=set_len_cv(3); const double len_cv_PH=set_len_cv(4);

  //COMBINED HL-LL FISHERY LANDINGS (power fct growth)
  const double agepar_a_F_LO=set_agepar_a_F(2); const double agepar_a_F_HI=set_agepar_a_F(3); const double agepar_a_F_PH=set_agepar_a_F(4);
  const double agepar_b_F_LO=set_agepar_b_F(2); const double agepar_b_F_HI=set_agepar_b_F(3); const double agepar_b_F_PH=set_agepar_b_F(4);
  const double len_cv_F_LO=set_len_cv_F(2); const double len_cv_F_HI=set_len_cv_F(3); const double len_cv_F_PH=set_len_cv_F(4);

  //MRIP FISHERY LANDINGS (power fct growth)
  const double agepar_a_mrip_LO=set_agepar_a_mrip(2); const double agepar_a_mrip_HI=set_agepar_a_mrip(3); const double agepar_a_mrip_PH=set_agepar_a_mrip(4);
  const double agepar_b_mrip_LO=set_agepar_b_mrip(2); const double agepar_b_mrip_HI=set_agepar_b_mrip(3); const double agepar_b_mrip_PH=set_agepar_b_mrip(4);
  const double len_cv_mrip_LO=set_len_cv_mrip(2); const double len_cv_mrip_HI=set_len_cv_mrip(3); const double len_cv_mrip_PH=set_len_cv_mrip(4);

  const double M_constant_LO=set_M_constant(2); const double M_constant_HI=set_M_constant(3); const double M_constant_PH=set_M_constant(4);

  const double steep_LO=set_steep(2); const double steep_HI=set_steep(3); const double steep_PH=set_steep(4);
  const double log_R0_LO=set_log_R0(2); const double log_R0_HI=set_log_R0(3); const double log_R0_PH=set_log_R0(4);
  const double R_autocorr_LO=set_R_autocorr(2); const double R_autocorr_HI=set_R_autocorr(3); const double R_autocorr_PH=set_R_autocorr(4);
  const double rec_sigma_LO=set_rec_sigma(2); const double rec_sigma_HI=set_rec_sigma(3); const double rec_sigma_PH=set_rec_sigma(4);

  const double selpar_L50_mrip_LO=set_selpar_L50_mrip(2); const double selpar_L50_mrip_HI=set_selpar_L50_mrip(3); const double selpar_L50_mrip_PH=set_selpar_L50_mrip(4);
  const double selpar_slope_mrip_LO=set_selpar_slope_mrip(2); const double selpar_slope_mrip_HI=set_selpar_slope_mrip(3); const double selpar_slope_mrip_PH=set_selpar_slope_mrip(4);
  const double selpar_L50_cHL_LO=set_selpar_L50_cHL(2); const double selpar_L50_cHL_HI=set_selpar_L50_cHL(3); const double selpar_L50_cHL_PH=set_selpar_L50_cHL(4);
  const double selpar_slope_cHL_LO=set_selpar_slope_cHL(2); const double selpar_slope_cHL_HI=set_selpar_slope_cHL(3); const double selpar_slope_cHL_PH=set_selpar_slope_cHL(4);
  const double selpar_L50_cLL_LO=set_selpar_L50_cLL(2); const double selpar_L50_cLL_HI=set_selpar_L50_cLL(3); const double selpar_L50_cLL_PH=set_selpar_L50_cLL(4);
  const double selpar_slope_cLL_LO=set_selpar_slope_cLL(2); const double selpar_slope_cLL_HI=set_selpar_slope_cLL(3); const double selpar_slope_cLL_PH=set_selpar_slope_cLL(4);

  const double log_q_cLL_LO=set_log_q_cLL(2); const double log_q_cLL_HI=set_log_q_cLL(3); const double log_q_cLL_PH=set_log_q_cLL(4);
  const double log_q_hb_LO=set_log_q_hb(2); const double log_q_hb_HI=set_log_q_hb(3); const double log_q_hb_PH=set_log_q_hb(4);
  const double log_q_cHL_LO=set_log_q_cHL(2); const double log_q_cHL_HI=set_log_q_cHL(3); const double log_q_cHL_PH=set_log_q_cHL(4);

  const double F_init_LO=set_F_init(2); const double F_init_HI=set_F_init(3); const double F_init_PH=set_F_init(4);
  const double log_avg_F_mrip_LO=set_log_avg_F_mrip(2); const double log_avg_F_mrip_HI=set_log_avg_F_mrip(3); const double log_avg_F_mrip_PH=set_log_avg_F_mrip(4);
  const double log_avg_F_cHL_LO=set_log_avg_F_cHL(2); const double log_avg_F_cHL_HI=set_log_avg_F_cHL(3); const double log_avg_F_cHL_PH=set_log_avg_F_cHL(4);
  const double log_avg_F_cLL_LO=set_log_avg_F_cLL(2); const double log_avg_F_cLL_HI=set_log_avg_F_cLL(3); const double log_avg_F_cLL_PH=set_log_avg_F_cLL(4);

  //-dev vectors-----------------------------------------------------------------------------------------------------
  const double log_F_dev_mrip_LO=set_log_F_dev_mrip(1); const double log_F_dev_mrip_HI=set_log_F_dev_mrip(2); const double log_F_dev_mrip_PH=set_log_F_dev_mrip(3);
  const double log_F_dev_cHL_LO=set_log_F_dev_cHL(1); const double log_F_dev_cHL_HI=set_log_F_dev_cHL(2); const double log_F_dev_cHL_PH=set_log_F_dev_cHL(3);
  const double log_F_dev_cLL_LO=set_log_F_dev_cLL(1); const double log_F_dev_cLL_HI=set_log_F_dev_cLL(2); const double log_F_dev_cLL_PH=set_log_F_dev_cLL(3);
  const double log_rec_dev_LO=set_log_rec_dev(1); const double log_rec_dev_HI=set_log_rec_dev(2); const double log_rec_dev_PH=set_log_rec_dev(3);
  const double log_Nage_dev_LO=set_log_Nage_dev(1); const double log_Nage_dev_HI=set_log_Nage_dev(2); const double log_Nage_dev_PH=set_log_Nage_dev(3);

 END_CALCS

////--------------Growth-------------------------------------------------------------------------
  //POPULATION GROWTH CURVE
  init_bounded_number Linf(Linf_LO,Linf_HI,Linf_PH);
  init_bounded_number K(K_LO,K_HI,K_PH);
  init_bounded_number t0(t0_LO,t0_HI,t0_PH);
  init_bounded_number len_cv_val(len_cv_LO,len_cv_HI,len_cv_PH);

  //COMBINED HL-LL LANDINGS GROWTH CURVE (power fct)
  init_bounded_number agepar_a_F(agepar_a_F_LO,agepar_a_F_HI,agepar_a_F_PH);
  init_bounded_number agepar_b_F(agepar_b_F_LO,agepar_b_F_HI,agepar_b_F_PH);
  init_bounded_number len_cv_val_F(len_cv_F_LO,len_cv_F_HI,len_cv_F_PH);

  //MRIP LANDINGS GROWTH CURVE (power fct)
  init_bounded_number agepar_a_mrip(agepar_a_mrip_LO,agepar_a_mrip_HI,agepar_a_mrip_PH);
  init_bounded_number agepar_b_mrip(agepar_b_mrip_LO,agepar_b_mrip_HI,agepar_b_mrip_PH);
  init_bounded_number len_cv_val_mrip(len_cv_mrip_LO,len_cv_mrip_HI,len_cv_mrip_PH);

  vector Linf_out(1,8);
  vector K_out(1,8);
  vector t0_out(1,8);
  vector len_cv_val_out(1,8);

  vector agepar_a_F_out(1,8);
```

```
  vector agepar_b_F_out(1,8);
  vector len_cv_val_F_out(1,8);

  vector agepar_a_mrip_out(1,8);
  vector agepar_b_mrip_out(1,8);
  vector len_cv_val_mrip_out(1,8);

  //These values used for intermediate and summary calcs as needed
  //POPULATION GROWTH CURVE
  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

  //These  values used for intermediate and summary calcs as needed
  //FISHERY LANDINGS GROWTH CURVE
  vector meanlen_FL_F(1,nages);   //mean fork length (mm) at age all fish--fishery
  vector wgt_g_F(1,nages);        //whole wgt in g
  vector wgt_kg_F(1,nages);       //whole wgt in kg of fishery
  vector wgt_mt_F(1,nages);
  vector wgt_klb_F(1,nages);      //whole wgt in 1000 lb of fishery
  vector wgt_lb_F(1,nages);       //whole wgt in lb of fishery

  vector meanlen_FL_mrip(1,nages); //mean fork length (mm) mrip fishery

  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_HL_mm(styr,endyr,1,nages);             // mean length at age of commercial HL landings in mm (may differ from popn mean)
  matrix wholewgt_HL_klb(styr,endyr,1,nages);    //whole wgt of commercial HL landings in 1000 lb
  matrix len_hb_mm(styr,endyr,1,nages);
  matrix wholewgt_hb_klb(styr,endyr,1,nages);
  matrix len_cHL_mm(styr,endyr,1,nages);
  matrix wholewgt_cHL_klb(styr,endyr,1,nages);
  matrix len_cLL_mm(styr,endyr,1,nages);
  matrix wholewgt_cLL_klb(styr,endyr,1,nages);

  //vector lbins(1,nlenbins);//NOT NEEDED, USE lenbins instead, after making it a vector instead of ivector

  matrix lenprob(1,nages,1,nlenbins);           //distn of size at age (age-length key, 3 cm bins) in population
  number zscore_len;                            //standardized normal values used for computing lenprob
  vector cprob_lenvec(1,nlenbins);              //cumulative probabilities used for computing lenprob
  number zscore_lzero;                          //standardized normal values for length = 0
  number cprob_lzero;                           //length probability mass below zero, used for computing lenprob

  //matrices below are used to match length comps
  matrix lenprob_mrip(1,nages,1,nlenbins);     //distn of size at age in mrip
  matrix lenprob_cHL(1,nages,1,nlenbins);    //distn of size at age in comm HL
  matrix lenprob_cLL(1,nages,1,nlenbins);    //distn of size at age in comm LL

  //POPULATION
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

  vector len_sd_F(1,nages);
  vector len_cv_F(1,nages); //for fishgraph

  vector len_sd_mrip(1,nages);
  vector len_cv_mrip(1,nages); //for fishgraph

////----Predicted length and age compositions
  matrix pred_mrip_lenc(1,nyr_mrip_lenc,1,nlenbins);
  matrix pred_cHL_lenc(1,nyr_cHL_lenc,1,nlenbins);
  matrix pred_cLL_lenc(1,nyr_cLL_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_cHL_agec(1,nyr_cHL_agec,1,nages);
  matrix ErrorFree_cHL_agec(1,nyr_cHL_agec,1,nages);
  matrix pred_cLL_agec(1,nyr_cLL_agec,1,nages);
  matrix ErrorFree_cLL_agec(1,nyr_cLL_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

//effective sample size applied in multinomial distributions
  vector nsamp_mrip_lenc_allyr(styr,endyr);
  vector nsamp_mrip_agec_allyr(styr,endyr);
  vector nsamp_cHL_lenc_allyr(styr,endyr);
  vector nsamp_cHL_agec_allyr(styr,endyr);
  vector nsamp_cLL_lenc_allyr(styr,endyr);
  vector nsamp_cLL_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_mrip_lenc_allyr(styr,endyr);
  vector nfish_cHL_lenc_allyr(styr,endyr);
  vector nfish_cLL_lenc_allyr(styr,endyr);
  vector nfish_mrip_agec_allyr(styr,endyr);
  vector nfish_cHL_agec_allyr(styr,endyr);
  vector nfish_cLL_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_mrip_lenc_allyr_out(styr,endyr);
  vector neff_cHL_lenc_allyr_out(styr,endyr);
  vector neff_cLL_lenc_allyr_out(styr,endyr);
  vector neff_mrip_agec_allyr_out(styr,endyr);
  vector neff_cHL_agec_allyr_out(styr,endyr);
  vector neff_cLL_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 active_f(1,nages);                 //Proportion of active spawners 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(log_R0_LO,log_R0_HI,log_R0_PH);        //log(virgin Recruitment)
  vector log_R0_out(1,8);
  //number log_R0;
  number R0;
                              //virgin recruitment
  init_bounded_number steep(steep_LO,steep_HI,steep_PH); //steepness
  vector steep_out(1,8);
  init_bounded_number rec_sigma(rec_sigma_LO,rec_sigma_HI,rec_sigma_PH);  //sd recruitment residuals
  vector rec_sigma_out(1,8);

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

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr_rec_dev,log_rec_dev_LO,log_rec_dev_HI,log_rec_dev_PH);  //log recruitment deviations

  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(R_autocorr_LO,R_autocorr_HI,R_autocorr_PH);  //autocorrelation in SR
  vector R_autocorr_out(1,8);

  init_bounded_dev_vector log_Nage_dev(2,nages,log_Nage_dev_LO,log_Nage_dev_HI,log_Nage_dev_PH);
  vector log_Nage_dev_out(2,nages);

  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(selpar_L50_mrip_LO,selpar_L50_mrip_HI,selpar_L50_mrip_PH);
  init_bounded_number selpar_slope_mrip(selpar_slope_mrip_LO,selpar_slope_mrip_HI,selpar_slope_mrip_PH);
  vector selpar_L50_mrip_out(1,8);
  vector selpar_slope_mrip_out(1,8);

//Commercial handline----------------------------------------------------
  matrix sel_cHL(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cHL(selpar_L50_cHL_LO,selpar_L50_cHL_HI,selpar_L50_cHL_PH);
  init_bounded_number selpar_slope_cHL(selpar_slope_cHL_LO,selpar_slope_cHL_HI,selpar_slope_cHL_PH);
  vector selpar_L50_cHL_out(1,8);
  vector selpar_slope_cHL_out(1,8);

//Commercial longline----------------------------------------------------
  matrix sel_cLL(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cLL(selpar_L50_cLL_LO,selpar_L50_cLL_HI,selpar_L50_cLL_PH);
  init_bounded_number selpar_slope_cLL(selpar_slope_cLL_LO,selpar_slope_cLL_HI,selpar_slope_cLL_PH);
  vector selpar_L50_cLL_out(1,8);
  vector selpar_slope_cLL_out(1,8);

//Headboat selectivity
  matrix sel_hb(styr,endyr,1,nages);

//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_cLL_cpue(styr_cLL_cpue,endyr_cLL_cpue);            //predicted comm LL U (pounds/hook
  matrix N_cLL(styr_cLL_cpue,endyr_cLL_cpue,1,nages);            //used to compute com LL 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_cHL_cpue(styr_cHL_cpue,endyr_cHL_cpue);                 //predicted com HL index
  matrix N_cHL(styr_cHL_cpue,endyr_cHL_cpue,1,nages);                 ////used to compute com HL index

//---Catchability (CPUE q's)-----------------------------------------------------
  init_bounded_number log_q_cLL(log_q_cLL_LO,log_q_cLL_HI,log_q_cLL_PH);
  init_bounded_number log_q_hb(log_q_hb_LO,log_q_hb_HI,log_q_hb_PH);
  init_bounded_number log_q_cHL(log_q_cHL_LO,log_q_cHL_HI,log_q_cHL_PH);
  vector log_q_cLL_out(1,8);
  vector log_q_hb_out(1,8);
  vector log_q_cHL_out(1,8);

  number q_rate;
  vector q_rate_fcn_cLL(styr_cLL_cpue,endyr_cLL_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_cHL(styr_cHL_cpue,endyr_cHL_cpue);         //increase due to technology creep (saturates in 2003)

  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_cLL(styr_cLL_cpue,endyr_cLL_cpue-1);
 vector q_RW_log_dev_hb(styr_hb_cpue,endyr_hb_cpue-1);
 vector q_RW_log_dev_cHL(styr_cHL_cpue,endyr_cHL_cpue-1);

 vector q_cLL(styr_cLL_cpue,endyr_cLL_cpue);
 vector q_hb(styr_hb_cpue,endyr_hb_cpue); //or number q_hb;
 vector q_cHL(styr_cHL_cpue,endyr_cHL_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_cHL_num(styr,endyr,1,nages);   //handline landings (numbers) at age
  matrix L_cHL_klb(styr,endyr,1,nages);   //handline landings (1000 lb whole weight) at age
  vector pred_cHL_L_knum(styr,endyr);     //yearly handline landings in 1000 fish summed over ages
  vector pred_cHL_L_klb(styr,endyr);      //yearly handline landings in 1000 lb summed over ages
  vector obs_cHL_L_wbias(styr,endyr);     //yearly handline landings observed, perhaps adjusted for multiplicative bias

  matrix L_cLL_num(styr,endyr,1,nages);   //LL landings (numbers) at age
  matrix L_cLL_klb(styr,endyr,1,nages);   //LL landings (1000 lb whole weight) at age
  vector pred_cLL_L_knum(styr,endyr);     //yearly LL landings in 1000 fish summed over ages
  vector pred_cLL_L_klb(styr,endyr);      //yearly LL landings in 1000 lb summed over ages
  vector obs_cLL_L_wbias(styr,endyr);     //yearly LL 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_cHL_prop;       //proportion of F_sum attributable to com HL, last X yr
  number F_cLL_prop;       //proportion of F_sum attributable to com LL, 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_spawn(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
  init_bounded_number M_constant(M_constant_LO,M_constant_HI,M_constant_PH);  //age-indpendent: used only for MSST
  vector M_constant_out(1,8);

  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
  matrix Z(styr,endyr,1,nages);

  init_bounded_number log_avg_F_mrip(log_avg_F_mrip_LO,log_avg_F_mrip_HI,log_avg_F_mrip_PH);
  vector log_avg_F_mrip_out(1,8);
  init_bounded_dev_vector log_F_dev_mrip(styr_mrip_L,endyr_mrip_L,log_F_dev_mrip_LO,log_F_dev_mrip_HI,log_F_dev_mrip_PH);
  vector log_F_dev_mrip_out(styr_mrip_L,endyr_mrip_L);
  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_cHL(log_avg_F_cHL_LO,log_avg_F_cHL_HI,log_avg_F_cHL_PH);
  vector log_avg_F_cHL_out(1,8);
  init_bounded_dev_vector log_F_dev_cHL(styr_cHL_L,endyr_cHL_L,log_F_dev_cHL_LO,log_F_dev_cHL_HI,log_F_dev_cHL_PH);
  vector log_F_dev_cHL_out(styr_cHL_L,endyr_cHL_L);
  matrix F_cHL(styr,endyr,1,nages);
  vector F_cHL_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cHL;
  number log_F_dev_end_cHL;

  init_bounded_number log_avg_F_cLL(log_avg_F_cLL_LO,log_avg_F_cLL_HI,log_avg_F_cLL_PH);
  vector log_avg_F_cLL_out(1,8);
  init_bounded_dev_vector log_F_dev_cLL(styr_cLL_L,endyr_cLL_L,log_F_dev_cLL_LO,log_F_dev_cLL_HI,log_F_dev_cLL_PH);
  vector log_F_dev_cLL_out(styr_cLL_L,endyr_cLL_L);
  matrix F_cLL(styr,endyr,1,nages);
  vector F_cLL_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cLL;
  number log_F_dev_end_cLL;

  init_bounded_number F_init(F_init_LO,F_init_HI,F_init_PH);
  vector F_init_out(1,8);

  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 recreational 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_spawn(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)

////-------SDNR output--------------------------------------------------------------------------------

  number sdnr_lc_mrip;
  number sdnr_lc_cHL;
  number sdnr_lc_cLL;

  number sdnr_ac_mrip;
  number sdnr_ac_cHL;
  number sdnr_ac_cLL;

  number sdnr_I_cLL;
  number sdnr_I_hb;
  number sdnr_I_cHL;

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

  number w_lc_mrip;
  number w_lc_cHL;
  number w_lc_cLL;

  number w_ac_mrip;
  number w_ac_cHL;
  number w_ac_cLL;

  number w_I_cLL;
  number w_I_hb;
  number w_I_cHL;

  number w_rec;
  number w_rec_early;
  number w_rec_end;
  number w_fullF;
  number w_Ftune;

  number f_cLL_cpue;
  number f_hb_cpue;
  number f_cHL_cpue;

  number f_mrip_L;
  number f_cHL_L;
  number f_cLL_L;

  number f_mrip_lenc;
  number f_cHL_lenc;
  number f_cLL_lenc;

  number f_mrip_agec;
  number f_cHL_agec;
  number f_cLL_agec;

  number f_cLL_RW_cpue; //random walk component of indices
  number f_hb_RW_cpue;   //random walk component of indices
  number f_cHL_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_priors;                   //prior information on parameters

  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)
  #include <time.h>
time_t start,finish;
long hour,minute,second;
double elapsed_time;

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
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(1);
  K=set_K(1);
  t0=set_t0(1);
  len_cv_val=set_len_cv(1);

//power fct fishery growth curve
  agepar_a_F=set_agepar_a_F(1); //combined HL-LL fishery
  agepar_b_F=set_agepar_b_F(1); //mrip fishery
  len_cv_val_F=set_len_cv_F(1);

//power fct fishery growth curve (mrip)
  agepar_a_mrip=set_agepar_a_mrip(1); //fishery
  agepar_b_mrip=set_agepar_b_mrip(1);       //fishery
  len_cv_val_mrip=set_len_cv_mrip(1);

  M=set_M;
  M_constant=set_M_constant(1);

  log_R0=set_log_R0(1);
  steep=set_steep(1);
  R_autocorr=set_R_autocorr(1);
  rec_sigma=set_rec_sigma(1);

  log_q_cLL=set_log_q_cLL(1);
  log_q_hb=set_log_q_hb(1);
  log_q_cHL=set_log_q_cHL(1);

  q_rate=set_q_rate;
  q_rate_fcn_cLL=1.0;
  q_rate_fcn_hb=1.0;
  q_rate_fcn_cHL=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;
  q_RW_log_dev_cLL.initialize();
  q_RW_log_dev_hb.initialize();
  q_RW_log_dev_cHL.initialize();


  if (set_q_rate_phase<0 & q_rate!=0.0)
  {
      for (iyear=styr_cLL_cpue; iyear<=endyr_cLL_cpue; iyear++)
      {   if (iyear>styr_cLL_cpue & iyear <=2003)
          {//q_rate_fcn_cLL(iyear)=(1.0+q_rate)*q_rate_fcn_cLL(iyear-1); //compound
             q_rate_fcn_cLL(iyear)=(1.0+(iyear-styr_cLL_cpue)*q_rate)*q_rate_fcn_cLL(styr_cLL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cLL(iyear)=q_rate_fcn_cLL(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_cHL_cpue; iyear<=endyr_cHL_cpue; iyear++)
      {   if (iyear>styr_cHL_cpue & iyear <=2003)
          {//q_rate_fcn_sc(iyear)=(1.0+q_rate)*q_rate_fcn_cHL(iyear-1); //compound
             q_rate_fcn_cHL(iyear)=(1.0+(iyear-styr_cHL_cpue)*q_rate)*q_rate_fcn_cHL(styr_cHL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cHL(iyear)=q_rate_fcn_cHL(iyear-1);}
```

```
      }


  } //end q_rate conditional

  w_L=set_w_L;

  w_lc_mrip=set_w_lc_mrip;
  w_lc_cHL=set_w_lc_cHL;
  w_lc_cLL=set_w_lc_cLL;

  w_ac_mrip=set_w_ac_mrip;
  w_ac_cHL=set_w_ac_cHL;
  w_ac_cLL=set_w_ac_cLL;

  w_I_cLL=set_w_I_cLL;
  w_I_hb=set_w_I_hb;
  w_I_cHL=set_w_I_cHL;

  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;

  log_avg_F_mrip=set_log_avg_F_mrip(1);
  log_avg_F_cHL=set_log_avg_F_cHL(1);
  log_avg_F_cLL=set_log_avg_F_cLL(1);

  log_F_dev_mrip=set_log_F_dev_mrip_vals;
  log_F_dev_cHL=set_log_F_dev_cHL_vals;
  log_F_dev_cLL=set_log_F_dev_cLL_vals;

  F_init=set_F_init(1);

  selpar_L50_mrip=set_selpar_L50_mrip(1);
  selpar_slope_mrip=set_selpar_slope_mrip(1);

  selpar_L50_cHL=set_selpar_L50_cHL(1);
  selpar_slope_cHL=set_selpar_slope_cHL(1);

  selpar_L50_cLL=set_selpar_L50_cLL(1);
  selpar_slope_cLL=set_selpar_slope_cLL(1);

  //cout << selpar_L50_cHL << selpar_slope_cHL << 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;
 active_f=active_f_obs;

 prop_f=prop_f_obs;

//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_cHL_lenc_allyr=missing;
       nsamp_cLL_lenc_allyr=missing;
       nsamp_mrip_agec_allyr=missing;
       nsamp_cHL_agec_allyr=missing;
       nsamp_cLL_agec_allyr=missing;

       nfish_mrip_lenc_allyr=missing;//"missing" defined in admb2r.cpp
       nfish_cHL_lenc_allyr=missing;
       nfish_cLL_lenc_allyr=missing;
       nfish_mrip_agec_allyr=missing;
       nfish_cHL_agec_allyr=missing;
       nfish_cLL_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_cHL_lenc; iyear++)
          {if (nsamp_cHL_lenc(iyear)>=minSS_cHL_lenc)
            {nsamp_cHL_lenc_allyr(yrs_cHL_lenc(iyear))=nsamp_cHL_lenc(iyear);
             nfish_cHL_lenc_allyr(yrs_cHL_lenc(iyear))=nfish_cHL_lenc(iyear);}}

       for (iyear=1; iyear<=nyr_cHL_agec; iyear++)
          {if (nsamp_cHL_agec(iyear)>=minSS_cHL_agec)
            {nsamp_cHL_agec_allyr(yrs_cHL_agec(iyear))=nsamp_cHL_agec(iyear);
             nfish_cHL_agec_allyr(yrs_cHL_agec(iyear))=nfish_cHL_agec(iyear);}}

       for (iyear=1; iyear<=nyr_cLL_lenc; iyear++)
          {if (nsamp_cLL_lenc(iyear)>=minSS_cLL_lenc)
```

```
          {nsamp_cLL_lenc_allyr(yrs_cLL_lenc(iyear))=nsamp_cLL_lenc(iyear);
           nfish_cLL_lenc_allyr(yrs_cLL_lenc(iyear))=nfish_cLL_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cLL_agec; iyear++)
         {if (nsamp_cLL_agec(iyear)>=minSS_cLL_agec)
           {nsamp_cLL_agec_allyr(yrs_cLL_agec(iyear))=nsamp_cLL_agec(iyear);
            nfish_cLL_agec_allyr(yrs_cLL_agec(iyear))=nfish_cLL_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_cHL.initialize();
  L_cHL_num.initialize();
  F_cLL.initialize();
  L_cLL_num.initialize();

  F_mrip_out.initialize();
  F_cHL_out.initialize();
  F_cLL_out.initialize();

  sel_mrip.initialize();
  sel_hb.initialize();
  sel_cHL.initialize();
  sel_cLL.initialize();

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

  log_rec_dev=set_log_rec_dev_vals;

  log_Nage_dev.initialize();


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
TOP_OF_MAIN_SECTION
  time(&start);
  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

 R0=mfexp(log_R0);

 //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
  //THESE CALCULATIONS ASSUME THE POPULATION GROWTH CURVE
```

```
  //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;
    wgt_klb=mt2klb*wgt_mt;                               //1000 lb of whole wgt
    wgt_lb=mt2lb*wgt_mt;                                 //lbs of whole wgt

    gonad_wgt_mt=g2mt*mfexp(gwgtpar_a+gwgtpar_b*(wgt_g));
    for (iage=1;iage<=nages;iage++)
    {
      if(gonad_wgt_mt(iage)<0){gonad_wgt_mt(iage)=0;}
    }

  //tHESE CALCULATIONS ASSUME THE FISHERY LANDINGS (COMBINED HL-LL) GROWTH CURVE
    meanlen_FL_F=agepar_a_F*pow(agebins,agepar_b_F);                  //fork length in mm fishery (power fct)
    wgt_kg_F=wgtpar_a*pow(meanlen_FL_F,wgtpar_b);            //fishery
    wgt_g_F=wgt_kg_F/g2kg;
    wgt_mt_F=wgt_g_F*g2mt;
    wgt_klb_F=mt2klb*wgt_mt_F;                               //fishery
    wgt_lb_F=mt2lb*wgt_mt_F;                                 //fishery

    //THESE CALCULATIONS ASSUME THE MRIP FISHERY GROWTH CURVE
    meanlen_FL_mrip=agepar_a_mrip*pow(agebins,agepar_b_mrip);

FUNCTION get_reprod
   reprod=elem_prod(elem_prod(prop_f,elem_prod(maturity_f,active_f)),wgt_mt);
   reprod2=elem_prod(elem_prod(prop_f,elem_prod(maturity_f,active_f)),wgt_mt);

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

  dvar_vector length(1,nages);
  dvariable cvlen;

  if (use_landings_growth==1.0)
      {
      length=meanlen_FL_F;
      cvlen=len_cv_val_F;
      }
  if  (use_landings_growth==0.0)
      {
      length=meanlen_FL;
      cvlen=len_cv_val;
      }

  for (iage=1;iage<=nages;iage++)
  {
    len_cv(iage)=cvlen;
    len_sd(iage)=length(iage)*len_cv(iage);

    zscore_lzero=(0.0-length(iage))/len_sd(iage);
    cprob_lzero=cumd_norm(zscore_lzero);

    //first length bin
    zscore_len=((lenbins(1)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
    cprob_lenvec(1)=cumd_norm(zscore_len);          //includes any probability mass below zero
    lenprob(iage,1)=cprob_lenvec(1)-cprob_lzero;    //removes any probability mass below zero

    //most other length bins
    for (ilen=2;ilen<nlenbins;ilen++)
      {
        zscore_len=((lenbins(ilen)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
        cprob_lenvec(ilen)=cumd_norm(zscore_len);
        lenprob(iage,ilen)=cprob_lenvec(ilen)-cprob_lenvec(ilen-1);
      }
    //last length bin is a plus group
    zscore_len=((lenbins(nlenbins)-0.5*lenbins_width)-length(iage)) / len_sd(iage);
    lenprob(iage,nlenbins)=1.0-cumd_norm(zscore_len);

    lenprob(iage)=lenprob(iage)/(1.0-cprob_lzero);  //renormalize to account for any prob mass below size=0
  }
  //fleet and survey specific length probs, all assumed here to equal the popn
  lenprob_cHL=lenprob;
  lenprob_cLL=lenprob;

 //This part to get the lenprob for mrip

  lenprob.initialize();
  length=meanlen_FL_mrip;
  cvlen=len_cv_val_mrip;

  //cout << length << cvlen << endl;

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

    zscore_lzero=(0.0-length(iage))/len_sd(iage);
    cprob_lzero=cumd_norm(zscore_lzero);

    //first length bin
    zscore_len=((lenbins(1)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
    cprob_lenvec(1)=cumd_norm(zscore_len);          //includes any probability mass below zero
    lenprob(iage,1)=cprob_lenvec(1)-cprob_lzero;    //removes any probability mass below zero

    //most other length bins
    for (ilen=2;ilen<nlenbins;ilen++)
      {
        zscore_len=((lenbins(ilen)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
        cprob_lenvec(ilen)=cumd_norm(zscore_len);
        lenprob(iage,ilen)=cprob_lenvec(ilen)-cprob_lenvec(ilen-1);
```

```
      }
    //last length bin is a plus group
    zscore_len=((lenbins(nlenbins)-0.5*lenbins_width)-length(iage)) / len_sd(iage);
    lenprob(iage,nlenbins)=1.0-cumd_norm(zscore_len);

    lenprob(iage)=lenprob(iage)/(1.0-cprob_lzero);  //renormalize to account for any prob mass below size=0
  }

  lenprob_mrip=lenprob;

FUNCTION get_weight_at_age_landings

  if(use_landings_growth==1.0){

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    len_mrip_mm(iyear)=meanlen_FL_F;
    wholewgt_mrip_klb(iyear)=wgt_klb_F;
    len_hb_mm(iyear)=meanlen_FL_F;
    wholewgt_hb_klb(iyear)=wgt_klb_F;
    len_cHL_mm(iyear)=meanlen_FL_F;
    wholewgt_cHL_klb(iyear)=wgt_klb_F;
    len_cLL_mm(iyear)=meanlen_FL_F;
    wholewgt_cLL_klb(iyear)=wgt_klb_F;
  }
  }

  if(use_landings_growth==0.0){

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    len_mrip_mm(iyear)=meanlen_FL;
    wholewgt_mrip_klb(iyear)=wgt_klb;
    len_hb_mm(iyear)=meanlen_FL;
    wholewgt_hb_klb(iyear)=wgt_klb;
    len_cHL_mm(iyear)=meanlen_FL;
    wholewgt_cHL_klb(iyear)=wgt_klb;
    len_cLL_mm(iyear)=meanlen_FL;
    wholewgt_cLL_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)*(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_cHL(iyear)=logistic(agebins, selpar_L50_cHL, selpar_slope_cHL);
     sel_cLL(iyear)=logistic(agebins, selpar_L50_cLL ,selpar_slope_cLL);
   }

  sel_hb=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_cHL=sum(log_F_dev_cHL(styr_cHL_L,(styr_cHL_L+2)))/3.0;
  log_F_dev_init_cLL=sum(log_F_dev_cLL(styr_cLL_L,(styr_cLL_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)); //}
       F_mrip(iyear)=sel_mrip(iyear)*F_mrip_out(iyear);
       Fsum(iyear)+=F_mrip_out(iyear);
    }

    if(iyear>=styr_cHL_L & iyear<=endyr_cHL_L)
    {  F_cHL_out(iyear)=mfexp(log_avg_F_cHL+log_F_dev_cHL(iyear)); //}
       F_cHL(iyear)=sel_cHL(iyear)*F_cHL_out(iyear);
       Fsum(iyear)+=F_cHL_out(iyear);
    }

    if(iyear>=styr_cLL_L & iyear<=endyr_cLL_L)
    {  F_cLL_out(iyear)=mfexp(log_avg_F_cLL+log_F_dev_cLL(iyear)); //}
       F_cLL(iyear)=sel_cLL(iyear)*F_cLL_out(iyear);
       Fsum(iyear)+=F_cLL_out(iyear);
    }

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

```
    F(iyear)+=F_cLL(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);

  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

  S0=spr_F0*R0;

  R_virgin=SR_eq_func(R0, steep, spr_F0, spr_F0, BiasCor, SR_switch);

  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;

  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

  spr_initial=sum(elem_prod(N_spr_initial,reprod));
  if (styr==styr_rec_dev) {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, 1.0, SR_switch);}
  else {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, BiasCor, SR_switch);} //with bias correction

  if(R1<10.0) {R1=10.0;} //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),reprod2)); //KC changed to reprod2
  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-1)) //recruitment follows S-R curve exactly
         N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch);
         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),reprod2));  //KC changed to reprod2
        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
    {

        //Ricker
        N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch)*mfexp(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),reprod2));  //KC changed to reprod2
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
    }
  }

    //Ricker
    N(endyr+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(endyr),SR_switch);
```

```
    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

//Time series of interest
  rec=column(N,1);
  SdSO=SSB/SO;

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_cHL_num(iyear,iage)=N(iyear,iage)*F_cHL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cLL_num(iyear,iage)=N(iyear,iage)*F_cLL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_mrip_L_knum(iyear)=sum(L_mrip_num(iyear))/1000.0;
    pred_cHL_L_knum(iyear)=sum(L_cHL_num(iyear))/1000.0;
    pred_cLL_L_knum(iyear)=sum(L_cLL_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_cHL_klb(iyear)=elem_prod(L_cHL_num(iyear),wholewgt_cHL_klb(iyear));     //in 1000 lb
    L_cLL_klb(iyear)=elem_prod(L_cLL_num(iyear),wholewgt_cLL_klb(iyear));     //in 1000 lb

    pred_mrip_L_klb(iyear)=sum(L_mrip_klb(iyear));
    pred_cHL_L_klb(iyear)=sum(L_cHL_klb(iyear));
    pred_cLL_L_klb(iyear)=sum(L_cLL_klb(iyear));
  }


FUNCTION get_catchability_fcns
 //Get rate increase if estimated, otherwise fixed above
  if (set_q_rate_phase>0.0)
  {
      for (iyear=styr_cLL_cpue; iyear<=endyr_cLL_cpue; iyear++)
      {   if (iyear>styr_cLL_cpue & iyear <=2003)
          {//q_rate_fcn_cLL(iyear)=(1.0+q_rate)*q_rate_fcn_cLL(iyear-1); //compound
             q_rate_fcn_cLL(iyear)=(1.0+(iyear-styr_cLL_cpue)*q_rate)*q_rate_fcn_cLL(styr_cLL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cLL(iyear)=q_rate_fcn_cLL(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_cHL_cpue; iyear<=endyr_cHL_cpue; iyear++)
      {   if (iyear>styr_cHL_cpue & iyear <=2003)
          {//q_rate_fcn_cHL(iyear)=(1.0+q_rate)*q_rate_fcn_cHL(iyear-1); //compound
             q_rate_fcn_cHL(iyear)=(1.0+(iyear-styr_cHL_cpue)*q_rate)*q_rate_fcn_cHL(styr_cHL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cHL(iyear)=q_rate_fcn_cHL(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------------------------

 //Rec HB cpue
  q_hb(styr_hb_cpue)=mfexp(log_q_hb);
  for (iyear=styr_hb_cpue; iyear<=endyr_hb_cpue; iyear++)
  {
      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));}
  }

 //Com LL cpue
  q_cLL(styr_cLL_cpue)=mfexp(log_q_cLL);
  for (iyear=styr_cLL_cpue; iyear<=endyr_cLL_cpue; iyear++)
  {
      N_cLL(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_cLL(iyear)),wholewgt_cLL_klb(iyear));
      pred_cLL_cpue(iyear)=q_cLL(iyear)*q_rate_fcn_cLL(iyear)*q_DD_fcn(iyear)*sum(N_cLL(iyear));
      if (iyear<endyr_cLL_cpue){q_cLL(iyear+1)=q_cLL(iyear)*mfexp(q_RW_log_dev_cLL(iyear));}
  }

 //Com HL cpue
  q_cHL(styr_cHL_cpue)=mfexp(log_q_cHL);
```

```
  for (iyear=styr_cHL_cpue; iyear<=endyr_cHL_cpue; iyear++)
  {
      N_cHL(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_cHL(iyear)),wholewgt_cHL_klb(iyear));
      pred_cHL_cpue(iyear)=q_cHL(iyear)*q_rate_fcn_cHL(iyear)*q_DD_fcn(iyear)*sum(N_cHL(iyear));
      if (iyear<endyr_cHL_cpue){q_cHL(iyear+1)=q_cHL(iyear)*mfexp(q_RW_log_dev_cHL(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)));
  }

 //Commercial HL
  for (iyear=1;iyear<=nyr_cHL_lenc;iyear++)
  {
        pred_cHL_lenc(iyear)=(L_cHL_num(yrs_cHL_lenc(iyear))*lenprob_cHL)
                        /sum(L_cHL_num(yrs_cHL_lenc(iyear)));
  }

 //Commercial LL
  for (iyear=1;iyear<=nyr_cLL_lenc;iyear++)
  {
        pred_cLL_lenc(iyear)=(L_cLL_num(yrs_cLL_lenc(iyear))*lenprob_cLL)
                        /sum(L_cLL_num(yrs_cLL_lenc(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);}

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

  //Commercial HL
  for (iyear=1;iyear<=nyr_cHL_agec;iyear++)
  {
    ErrorFree_cHL_agec(iyear)=elem_prod(N(yrs_cHL_agec(iyear)),sel_cHL(yrs_cHL_agec(iyear)));
    pred_cHL_agec(iyear)=age_error*(ErrorFree_cHL_agec(iyear)/sum(ErrorFree_cHL_agec(iyear)));
  }
   //Commercial LL
  for (iyear=1;iyear<=nyr_cLL_agec;iyear++)
  {
    ErrorFree_cLL_agec(iyear)=elem_prod(N(yrs_cLL_agec(iyear)),sel_cLL(yrs_cLL_agec(iyear)));
    pred_cLL_agec(iyear)=age_error*(ErrorFree_cLL_agec(iyear)/sum(ErrorFree_cLL_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_cHL+
        sum(log_F_dev_cHL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cLL+
        sum(log_F_dev_cLL((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_cHL_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cHL+
        sum(log_F_dev_cHL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cLL_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cLL+
        sum(log_F_dev_cLL((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_cHL=sum(log_F_dev_cHL((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_cLL=sum(log_F_dev_cLL((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_cHL(endyr)*mfexp(log_avg_F_cHL+log_F_dev_end_cHL)+
          sel_cLL(endyr)*mfexp(log_avg_F_cLL+log_F_dev_end_cLL);

  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_cHL_prop+F_cLL_prop;
  wgt_wgted_L_klb=F_mrip_prop/wgt_wgted_L_denom*wholewgt_mrip_klb(endyr)+
                  F_cHL_prop/wgt_wgted_L_denom*wholewgt_cHL_klb(endyr)+
                  F_cLL_prop/wgt_wgted_L_denom*wholewgt_cLL_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_spawn(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_spawn(nages)=(N_age_msy_spawn(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_spawn,reprod));
      //Ricker

    R_eq(ff)=SR_eq_func(R0, steep, spr_msy(1), spr_msy(ff), BiasCor, SR_switch);

    if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
    N_age_msy*=R_eq(ff);
    N_age_msy_spawn*=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_spawn,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);}

    for(iage=1;iage<=nages;iage++)
     {
     len_cv(iage)=len_cv_val;
     len_sd(iage)=meanlen_FL(iage)*len_cv(iage);
     len_cv_F(iage)=len_cv_val_F;
len_sd_F(iage)=meanlen_FL_F(iage)*len_cv_F(iage);
        len_cv_mrip(iage)=len_cv_val_mrip;
        len_sd_mrip(iage)=meanlen_FL_mrip(iage)*len_cv_mrip(iage);
   }

  //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_cHL_L_klb(iyear)+pred_cLL_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_mrip_L_knum(iyear)+pred_cHL_L_knum(iyear)+pred_cLL_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_cHL_num+L_cLL_num;   //landings at age in number fish
  L_total_klb=L_mrip_klb+L_cHL_klb+L_cLL_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));

  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(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_spawn(1,(nages-1))=elem_prod(N_age_spr(1,(nages-1)),
                                mfexp(-1.*Z(iyear)(1,(nages-1))*spawn_time_frac));
    N_age_spr_spawn(nages)=(N_age_spr_spawn(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_spawn,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_spawn(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_spawn(nages)=(N_age_spr_spawn(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_spawn,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_cHL_lenc_allyr_out=missing;
      neff_cLL_lenc_allyr_out=missing;
      neff_mrip_agec_allyr_out=missing;
      neff_cHL_agec_allyr_out=missing;
      neff_cLL_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_cHL_lenc; iyear++)
         {if (nsamp_cHL_lenc(iyear)>=minSS_cHL_lenc)
            {neff_cHL_lenc_allyr_out(yrs_cHL_lenc(iyear))=multinom_eff_N(pred_cHL_lenc(iyear),obs_cHL_lenc(iyear));}
          else {neff_cHL_lenc_allyr_out(yrs_cHL_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cLL_lenc; iyear++)
         {if (nsamp_cLL_lenc(iyear)>=minSS_cLL_lenc)
            {neff_cLL_lenc_allyr_out(yrs_cLL_lenc(iyear))=multinom_eff_N(pred_cLL_lenc(iyear),obs_cLL_lenc(iyear));}
          else {neff_cLL_lenc_allyr_out(yrs_cLL_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_cHL_agec; iyear++)
         {if (nsamp_cHL_agec(iyear)>=minSS_cHL_agec)
            {neff_cHL_agec_allyr_out(yrs_cHL_agec(iyear))=multinom_eff_N(pred_cHL_agec(iyear),obs_cHL_agec(iyear));}
          else {neff_cHL_agec_allyr_out(yrs_cHL_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cLL_agec; iyear++)
```

```
        {if (nsamp_cLL_agec(iyear)>=minSS_cLL_agec)
           {neff_cLL_agec_allyr_out(yrs_cLL_agec(iyear))=multinom_eff_N(pred_cLL_agec(iyear),obs_cLL_agec(iyear));}
         else {neff_cLL_agec_allyr_out(yrs_cLL_agec(iyear))=-99;}
        }
//--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

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

  f_cLL_cpue=0.0;
  f_cLL_cpue=lk_lognormal(pred_cLL_cpue, obs_cLL_cpue, cLL_cpue_cv, w_I_cLL);
  fval+=f_cLL_cpue;
  fval_data+=f_cLL_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_cHL_cpue=0.0;
  f_cHL_cpue=lk_lognormal(pred_cHL_cpue, obs_cHL_cpue, cHL_cpue_cv, w_I_cHL);
  fval+=f_cHL_cpue;
  fval_data+=f_cHL_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);
  fval+=f_mrip_L;
  fval_data+=f_mrip_L;

  //f_cHL_L in 1000 lb wholewgt
  f_cHL_L=lk_lognormal(pred_cHL_L_klb(styr_cHL_L,endyr_cHL_L), obs_cHL_L(styr_cHL_L,endyr_cHL_L),
                        cHL_L_cv(styr_cHL_L,endyr_cHL_L), w_L);
  fval+=f_cHL_L;
  fval_data+=f_cHL_L;

  //f_cLL_L in 1000 lb wholewgt
  f_cLL_L=lk_lognormal(pred_cLL_L_klb(styr_cLL_L,endyr_cLL_L), obs_cLL_L(styr_cLL_L,endyr_cLL_L),
                        cLL_L_cv(styr_cLL_L,endyr_cLL_L), w_L);
  fval+=f_cLL_L;
  fval_data+=f_cLL_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_cHL_lenc
  f_cHL_lenc=lk_robust_multinomial(nsamp_cHL_lenc, pred_cHL_lenc, obs_cHL_lenc, nyr_cHL_lenc, double(nlenbins), minSS_cHL_lenc, w_lc_cHL);
  //f_cHL_lenc=lk_multinomial(nsamp_cHL_lenc, pred_cHL_lenc, obs_cHL_lenc, nyr_cHL_lenc, minSS_cHL_lenc, w_lc_cHL);
  //fval+=f_cHL_lenc;
  //fval_data+=f_cHL_lenc;

  //f_cLL_lenc
  f_cLL_lenc=lk_robust_multinomial(nsamp_cLL_lenc, pred_cLL_lenc, obs_cLL_lenc, nyr_cLL_lenc, double(nlenbins), minSS_cLL_lenc, w_lc_cLL);
  //f_cLL_lenc=lk_multinomial(nsamp_cLL_lenc, pred_cLL_lenc, obs_cLL_lenc, nyr_cLL_lenc, minSS_cLL_lenc, w_lc_cLL);
  //fval+=f_cLL_lenc;
  //fval_data+=f_cLL_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_cHL_agec
  f_cHL_agec=lk_robust_multinomial(nsamp_cHL_agec, pred_cHL_agec, obs_cHL_agec, nyr_cHL_agec, double(nages), minSS_cHL_agec, w_ac_cHL);
  //f_cHL_agec=lk_multinomial(nsamp_cHL_agec, pred_cHL_agec, obs_cHL_agec, nyr_cHL_agec, minSS_cHL_agec, w_ac_cHL);
  fval+=f_cHL_agec;
  fval_data+=f_cHL_agec;

  //f_cLL_agec
  f_cLL_agec=lk_robust_multinomial(nsamp_cLL_agec, pred_cLL_agec, obs_cLL_agec, nyr_cLL_agec, double(nages), minSS_cLL_agec, w_ac_cLL);
  //f_cLL_agec=lk_multinomial(nsamp_cLL_agec, pred_cLL_agec, obs_cLL_agec, nyr_cLL_agec, minSS_cLL_agec, w_ac_cLL);
  fval+=f_cLL_agec;
  fval_data+=f_cLL_agec;

////----------Constraints and penalties--------------------------------
  f_rec_dev=0.0;
  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_rec_dev; 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));}
```

```
      }
  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_rec_dev)
      {
        for(iyear=(endyr_rec_phase2+1); iyear<=endyr_rec_dev; iyear++)
        {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_cLL_RW_cpue=0.0;
  for (iyear=styr_cLL_cpue; iyear<endyr_cLL_cpue; iyear++)
      {f_cLL_RW_cpue+=square(q_RW_log_dev_cLL(iyear))/(2.0*set_q_RW_cLL_var);}
  fval+=f_cLL_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_cHL_RW_cpue=0.0;
  for (iyear=styr_cHL_cpue; iyear<endyr_cHL_cpue; iyear++)
      {f_cHL_RW_cpue+=square(q_RW_log_dev_cHL(iyear))/(2.0*set_q_RW_cHL_var);}
  fval+=f_cHL_RW_cpue;


//---Priors-----------------------------------------------------
//neg_log_prior arguments: estimate, prior mean, prior var/-CV, 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(Linf,set_Linf(5),set_Linf(6),set_Linf(7));
  f_priors+=neg_log_prior(K,set_K(5),set_K(6),set_K(7));
  f_priors+=neg_log_prior(t0,set_t0(5),set_t0(6),set_t0(7));
  f_priors+=neg_log_prior(len_cv_val,set_len_cv(5),set_len_cv(6),set_len_cv(7));

  f_priors+=neg_log_prior(agepar_a_F,set_agepar_a_F(5),set_agepar_a_F(6),set_agepar_a_F(7));
  f_priors+=neg_log_prior(agepar_b_F,set_agepar_b_F(5),set_agepar_b_F(6),set_agepar_b_F(7));
  f_priors+=neg_log_prior(len_cv_val_F,set_len_cv_F(5),set_len_cv_F(6),set_len_cv_F(7));

  f_priors+=neg_log_prior(agepar_a_mrip,set_agepar_a_mrip(5),set_agepar_a_mrip(6),set_agepar_a_mrip(7));
  f_priors+=neg_log_prior(agepar_b_mrip,set_agepar_b_mrip(5),set_agepar_b_mrip(6),set_agepar_b_mrip(7));
  f_priors+=neg_log_prior(len_cv_val_mrip,set_len_cv_mrip(5),set_len_cv_mrip(6),set_len_cv_mrip(7));

  f_priors+=neg_log_prior(M_constant,set_M_constant(5),set_M_constant(6),set_M_constant(7));

  f_priors+=neg_log_prior(steep,set_steep(5),set_log_R0(6),set_log_R0(7));
  f_priors+=neg_log_prior(log_R0,set_log_R0(5),set_log_R0(6),set_log_R0(7));
  f_priors+=neg_log_prior(R_autocorr,set_R_autocorr(5),set_R_autocorr(6),set_R_autocorr(7));
  f_priors+=neg_log_prior(rec_sigma,set_rec_sigma(5),set_rec_sigma(6),set_rec_sigma(7));

  f_priors+=neg_log_prior(selpar_L50_mrip,set_selpar_L50_mrip(5),set_selpar_L50_mrip(6), set_selpar_L50_mrip(7));
  f_priors+=neg_log_prior(selpar_slope_mrip,set_selpar_slope_mrip(5),set_selpar_slope_mrip(6), set_selpar_slope_mrip(7));
  f_priors+=neg_log_prior(selpar_L50_cHL,set_selpar_L50_cHL(5), set_selpar_L50_cHL(6), set_selpar_L50_cHL(7));
  f_priors+=neg_log_prior(selpar_slope_cHL,set_selpar_slope_cHL(5), set_selpar_slope_cHL(6), set_selpar_slope_cHL(7));
  f_priors+=neg_log_prior(selpar_L50_cLL,set_selpar_L50_cLL(5), set_selpar_L50_cLL(6), set_selpar_L50_cLL(7));
  f_priors+=neg_log_prior(selpar_slope_cLL,set_selpar_slope_cLL(5), set_selpar_slope_cLL(6), set_selpar_slope_cLL(7));

  f_priors+=neg_log_prior(log_q_cLL,set_log_q_cLL(5),set_log_q_cLL(6),set_log_q_cLL(7));
  f_priors+=neg_log_prior(log_q_hb,set_log_q_hb(5),set_log_q_hb(6),set_log_q_hb(7));
  f_priors+=neg_log_prior(log_q_cHL,set_log_q_cHL(5),set_log_q_cHL(6),set_log_q_cHL(7));

  f_priors+=neg_log_prior(F_init,set_F_init(5),set_F_init(6),set_F_init(7));
  f_priors+=neg_log_prior(log_avg_F_mrip,set_log_avg_F_mrip(5),set_log_avg_F_mrip(6),set_log_avg_F_mrip(7));
  f_priors+=neg_log_prior(log_avg_F_cHL,set_log_avg_F_cHL(5),set_log_avg_F_cHL(6),set_log_avg_F_cHL(7));
  f_priors+=neg_log_prior(log_avg_F_cLL,set_log_avg_F_cLL(5),set_log_avg_F_cLL(6),set_log_avg_F_cLL(7));

  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-exponential: 4 parameters (but 1 is fixed)
FUNCTION dvar_vector logistic_exponential(const dvar_vector& ages, const dvariable& L50, const dvariable& slope, const dvariable& sigma, const dvariable& joint)
  //ages=vector of ages, L50=age at 50% sel (ascending limb), slope=rate of increase, sigma=controls rate of descent (descending)
  //joint=age to join curves
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1.0;
  for (iage=1; iage<=nages; iage++)
  {
   if (ages(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope*(ages(iage)-L50)));}
   if (ages(iage)>joint){Sel_Tmp(iage)=mfexp(-1.*square((ages(iage)-joint)/sigma));}
  }
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  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 or Ricker)
FUNCTION dvariable SR_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const dvariable& SSB, int func)
  //R0=virgin recruitment, h=steepness, spr_F0=spawners per recruit @ F=0, SSB=spawning biomass
  //func=1 for Beverton-Holt, 2 for Ricker
  RETURN_ARRAYS_INCREMENT();
  dvariable Recruits_Tmp;
  switch(func) {
    case 1: //Beverton-Holt
      Recruits_Tmp=((0.8*R0*h*SSB)/(0.2*R0*spr_F0*(1.0-h)+(h-0.2)*SSB));
    break;
    case 2: //Ricker
      Recruits_Tmp=((SSB/spr_F0)*mfexp(h*(1-SSB/(R0*spr_F0))));
    break;
  }
  RETURN_ARRAYS_DECREMENT();
  return Recruits_Tmp;

//--------------------------------------------------------------------------------
//Spawner-recruit equilibrium function (Beverton-Holt or Ricker)
FUNCTION dvariable SR_eq_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const dvariable& spr_F, const dvariable& BC, int func)
  //R0=virgin recruitment, h=steepness, spr_F0=spawners per recruit @ F=0, spr_F=spawners per recruit @ F, BC=bias correction
  //func=1 for Beverton-Holt, 2 for Ricker
  RETURN_ARRAYS_INCREMENT();
  dvariable Recruits_Tmp;
  switch(func) {
    case 1: //Beverton-Holt
      Recruits_Tmp=(R0/((5.0*h-1.0)*spr_F))*(BC*4.0*h*spr_F-spr_F0*(1.0-h));
    break;
    case 2: //Ricker
      Recruits_Tmp=R0/(spr_F/spr_F0)*(1.0+log(BC*spr_F/spr_F0)/h);
    break;
  }
  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=0.00001;
  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_dat)
  //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;

//--------------------------------------------------------------------------------
//SDNR: age comp likelihood (assumes fits are done with the robust multinomial function)
FUNCTION dvariable sdnr_multinomial(const double& ncomp, const dvar_vector& ages, const dvar_vector& nsamp,
                                const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const dvariable& wgt_dat)
  //ncomp=number of years of data, ages=vector of ages, nsamp=vector of N's,
  //pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, wgt_dat=likelihood weight for data source
  RETURN_ARRAYS_INCREMENT();
  dvariable SdnrTmp;
  dvar_vector o(1,ncomp);
  dvar_vector p(1,ncomp);
  dvar_vector ose(1,ncomp);
  dvar_vector res(1,ncomp);
  SdnrTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {
    o(ii)=sum(elem_prod(ages,obs_comp(ii)));
    p(ii)=sum(elem_prod(ages,pred_comp(ii)));
    ose(ii)=sqrt((sum(elem_prod(square(ages),pred_comp(ii)))-square(p(ii)))/(nsamp(ii)*wgt_dat));
  }
  res=elem_div((o-p),ose);
  SdnrTmp=sqrt(sum(square(res-(sum(res)/ncomp))/(ncomp-1.0)));
  RETURN_ARRAYS_DECREMENT();
  return SdnrTmp;


//--------------------------------------------------------------------------------
//SDNR: lognormal likelihood
FUNCTION dvariable sdnr_lognormal(const dvar_vector& pred, const dvar_vector& obs, const dvar_vector& cv, const dvariable& wgt_dat)
  //nyr=number of years of data, pred=vector of predicted data, obs=vector of observed data, cv=vector of cv's, wgt_dat=likelihood weight for data source
  RETURN_ARRAYS_INCREMENT();
  dvariable SdnrTmp;
  dvariable small_number=0.00001;
  dvariable n;
  dvar_vector res(cv.indexmin(),cv.indexmax());
  SdnrTmp=0.0;
  res=elem_div(log(elem_div(obs+small_number,pred+small_number)),sqrt(log(1+square(cv/wgt_dat))));
  n=cv.indexmax()-cv.indexmin()+1;
  SdnrTmp=sqrt(sum(square(res-(sum(res)/n))/(n-1.0)));
  RETURN_ARRAYS_DECREMENT();
  return SdnrTmp;

//--------------------------------------------------------------------------------
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();

      time(&finish);
  elapsed_time=difftime(finish,start);
  hour=long(elapsed_time)/3600;
  minute=long(elapsed_time)%3600/60;
  second=(long(elapsed_time)%3600)%60;
  cout<<endl<<endl<<"*****************************************"<<endl;
  cout<<"--Start time: "<<ctime(&start)<<endl;
  cout<<"--Finish time: "<<ctime(&finish)<<endl;
  cout<<"--Runtime: ";
  cout<<hour<<" hours, "<<minute<<" minutes, "<<second<<" seconds"<<endl;
  cout<<"*****************************************"<<endl;

      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<<"   R0="<<R0<<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;

      sdnr_lc_mrip=sdnr_multinomial(nyr_mrip_lenc, lenbins, nsamp_mrip_lenc, pred_mrip_lenc, obs_mrip_lenc, w_lc_mrip);
      sdnr_lc_cHL=sdnr_multinomial(nyr_cHL_lenc, lenbins, nsamp_cHL_lenc, pred_cHL_lenc, obs_cHL_lenc, w_lc_cHL);
      sdnr_lc_cLL=sdnr_multinomial(nyr_cLL_lenc, lenbins, nsamp_cLL_lenc, pred_cLL_lenc, obs_cLL_lenc, w_lc_cLL);

      sdnr_ac_mrip=sdnr_multinomial(nyr_mrip_agec, agebins, nsamp_mrip_agec, pred_mrip_agec, obs_mrip_agec, w_ac_mrip);
      sdnr_ac_cHL=sdnr_multinomial(nyr_cHL_agec, agebins, nsamp_cHL_agec, pred_cHL_agec, obs_cHL_agec, w_ac_cHL);
      sdnr_ac_cLL=sdnr_multinomial(nyr_cLL_agec, agebins, nsamp_cLL_agec, pred_cLL_agec, obs_cLL_agec, w_ac_cLL);

      sdnr_I_cLL=sdnr_lognormal(pred_cLL_cpue, obs_cLL_cpue, cLL_cpue_cv, w_I_cLL);
      sdnr_I_hb=sdnr_lognormal(pred_hb_cpue, obs_hb_cpue, hb_cpue_cv, w_I_hb);
      sdnr_I_cHL=sdnr_lognormal(pred_cHL_cpue, obs_cHL_cpue, cHL_cpue_cv, w_I_cHL);
```

```
    //#############################################################################################
    //##  Passing parameters to vector for bounds check plotting
    //#############################################################################################
     Linf_out(8)=Linf; Linf_out(1,7)=set_Linf;
     K_out(8)=K; K_out(1,7)=set_K;
     t0_out(8)=t0; t0_out(1,7)=set_t0;
     len_cv_val_out(8)=len_cv_val; len_cv_val_out(1,7)=set_len_cv;

     agepar_a_F_out(8)=agepar_a_F; agepar_a_F_out(1,7)=set_agepar_a_F;
     agepar_b_F_out(8)=agepar_b_F; agepar_b_F_out(1,7)=set_agepar_b_F;
     len_cv_val_F_out(8)=len_cv_val_F; len_cv_val_F_out(1,7)=set_len_cv_F;

     agepar_a_mrip_out(8)=agepar_a_mrip; agepar_a_mrip_out(1,7)=set_agepar_a_mrip;
     agepar_b_mrip_out(8)=agepar_b_mrip; agepar_b_mrip_out(1,7)=set_agepar_b_mrip;
     len_cv_val_mrip_out(8)=len_cv_val_mrip; len_cv_val_mrip_out(1,7)=set_len_cv_mrip;

     log_R0_out(8)=log_R0; log_R0_out(1,7)=set_log_R0;
     steep_out(8)=steep; steep_out(1,7)=set_steep;
     rec_sigma_out(8)=rec_sigma; rec_sigma_out(1,7)=set_rec_sigma;
     R_autocorr_out(8)=R_autocorr; R_autocorr_out(1,7)=set_R_autocorr;

     selpar_L50_mrip_out(8)=selpar_L50_mrip; selpar_L50_mrip_out(1,7)=set_selpar_L50_mrip;
     selpar_slope_mrip_out(8)=selpar_slope_mrip; selpar_slope_mrip_out(1,7)=set_selpar_slope_mrip;
     selpar_L50_cHL_out(8)=selpar_L50_cHL; selpar_L50_cHL_out(1,7)=set_selpar_L50_cHL;
     selpar_slope_cHL_out(8)=selpar_slope_cHL; selpar_slope_cHL_out(1,7)=set_selpar_slope_cHL;
     selpar_L50_cLL_out(8)=selpar_L50_cLL; selpar_L50_cLL_out(1,7)=set_selpar_L50_cLL;
     selpar_slope_cLL_out(8)=selpar_slope_cLL; selpar_slope_cLL_out(1,7)=set_selpar_slope_cLL;

     log_q_cLL_out(8)=log_q_cLL; log_q_cLL_out(1,7)=set_log_q_cLL;
     log_q_hb_out(8)=log_q_hb; log_q_hb_out(1,7)=set_log_q_hb;
     log_q_cHL_out(8)=log_q_cHL; log_q_cHL_out(1,7)=set_log_q_cHL;

     M_constant_out(8)=M_constant; M_constant_out(1,7)=set_M_constant;

     log_avg_F_mrip_out(8)=log_avg_F_mrip; log_avg_F_mrip_out(1,7)=set_log_avg_F_mrip;
     log_avg_F_cHL_out(8)=log_avg_F_cHL; log_avg_F_cHL_out(1,7)=set_log_avg_F_cHL;
     log_avg_F_cLL_out(8)=log_avg_F_cLL; log_avg_F_cLL_out(1,7)=set_log_avg_F_cLL;
     F_init_out(8)=F_init; F_init_out(1,7)=set_F_init;

     log_rec_dev_output(styr_rec_dev,endyr_rec_dev)=log_rec_dev;
     log_F_dev_mrip_out(styr_mrip_L,endyr_mrip_L)=log_F_dev_mrip;
     log_F_dev_cHL_out(styr_cHL_L,endyr_cHL_L)=log_F_dev_cHL;
     log_F_dev_cLL_out(styr_cLL_L,endyr_cLL_L)=log_F_dev_cLL;

     log_Nage_dev_out(2,nages)=log_Nage_dev;


  #include "blt-baserevised2_make_Robject.cxx"   // write the S-compatible report

} //endl last phase loop
```

Appendix B Data inputs for the Beaufort Assessment Model

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  SEDAR 32 Blueline tilefish assessment May 2013
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: set-up section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
#Starting and ending year of model
1974
2011
##
#Starting year to estimate recruitment deviation from S-R curve
1974
##
#Ending year to estimate recruitment deviation from S-R curve (if change this remember to change initialization of recruit dev vector)
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
1990
2010
##
#Ending year for first selectivity period
2011
##
#Number of ages (15 classes is 1,...,15+) //assumes 15+ is plus group
15
#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 13.0 14.0 15.0
##
#Number length bins used to match length comps and width of bins
16
30.0
##
#Vector of length bins (mm)(midpoint of bin) used to match length comps
260.0 290.0 320.0 350.0 380.0 410.0 440.0 470.0 500.0 530.0 560.0 590.0 620.0 650.0 680.0 710.0
##
#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
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: observed data section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
#####################Recreational Landings##############################################################
##
#Total recreational landings (includes MRFSS landings+discards, SRHS landings+discards (assume 100% discard mortality rate)
#Starting and ending years of landings time series, respectively
1974
2011
#Observed total recreational removals (1000s of fish) and assumed CVs (MRFSS + SRHS + dead discards)
3.870   1.790   3.555   1.434   1.641   0.407   4.081   1.621   3.914   3.835   2.877   0.649   0.679   2.879   0.436   0.588   0.209   3.875   1.722   2.865   0.098   5.495   3.269  15.931   0.120
1.666   0.117   4.953   0.698   8.595   2.725  12.738  57.825 120.177  74.068   26.678   15.724   12.637
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    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05     0.05     0.05     0.05     0.05
##
#Number and vector of years of length compositions for recreational fishery (MRFSS)
25
1974  1975  1976  1977  1978  1979  1980  1981  1982  1983  1984  1986  1996  1997  1998  1999  2000  2001  2003  2006  2007  2008  2009  2010  2011
#sample size of recreational length comp data by year (first row observed Ntrips, second row Nfish)
27.0  29.0  44.0   18.0  14.0  10.0  18.0  12.0  9.0  19.0  13.0  11.0   7.0  14.0   9.0   8.0   6.0   5.0  12.0   17.0   34.0   46.0   28.0   29.0   18.0
91.0  78.0  187.0  66.0  32.0  61.0  45.0  36.0 18.0  43.0  29.0  30.0  50.0  81.0  36.0  19.0  39.0  34.0  41.0  165.0  358.0  406.0  230.0  415.0  678.0
##
#recreational length composition samples (year, lengthbin 3 cm; weighted)
0.000000    0.000000    0.000000    0.011111    0.000000    0.088889    0.022222    0.100000    0.166667    0.122222    0.088889    0.055556    0.144444    0.088889    0.077778    0.033333
0.000000    0.000000    0.012987    0.000000    0.025974    0.038961    0.090909    0.077922    0.090909    0.090909    0.077922    0.142857    0.090909    0.129870    0.103896    0.025974
0.000000    0.000000    0.000000    0.000000    0.010753    0.021505    0.075269    0.112903    0.139785    0.166667    0.155914    0.112903    0.123656    0.037634    0.037634    0.005376
0.000000    0.000000    0.000000    0.000000    0.015385    0.046154    0.107692    0.107692    0.184615    0.107692    0.107692    0.153846    0.046154    0.092308    0.015385    0.015385
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.032258    0.064516    0.129032    0.161290    0.161290    0.161290    0.193548    0.064516    0.032258
0.000000    0.049180    0.098361    0.081967    0.180328    0.131148    0.016393    0.032787    0.065574    0.098361    0.098361    0.032787    0.016393    0.049180    0.049180    0.000000
0.000000    0.000000    0.000000    0.088889    0.088889    0.155556    0.088889    0.022222    0.088889    0.155556    0.111111    0.044444    0.066667    0.066667    0.022222    0.000000
0.000000    0.000000    0.000000    0.028571    0.228571    0.114286    0.114286    0.085714    0.057143    0.085714    0.085714    0.057143    0.028571    0.057143    0.028571    0.028571
0.000000    0.000000    0.055556    0.000000    0.000000    0.111111    0.055556    0.388889    0.055556    0.055556    0.166667    0.111111    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.046512    0.023256    0.023256    0.046512    0.046512    0.069767    0.116279    0.255814    0.116279    0.186047    0.000000    0.069767    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.103448    0.103448    0.103448    0.137931    0.137931    0.137931    0.068966    0.034483    0.034483    0.068966    0.034483    0.034483
0.000000    0.000000    0.000000    0.100000    0.133333    0.066667    0.100000    0.133333    0.100000    0.100000    0.133333    0.000000    0.066667    0.066667    0.000000    0.000000
0.000000    0.000000    0.033880    0.000000    0.017740    0.120982    0.069361    0.053221    0.088702    0.191943    0.177404    0.104842    0.070962    0.053221    0.017740    0.000000
0.000339    0.000339    0.000452    0.049944    0.000169    0.149718    0.000169    0.049944    0.299492    0.149774    0.100056    0.000056    0.049944    0.049887    0.099718    0.000000
0.000000    0.083333    0.000000    0.083333    0.000000    0.138889    0.083333    0.111111    0.111111    0.194444    0.138889    0.027778    0.027778    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.055556    0.222222    0.055556    0.166667    0.055556    0.000000    0.000000    0.166667    0.055556    0.111111    0.055556    0.055556    0.000000
0.004185    0.008369    0.025108    0.014646    0.006277    0.006277    0.004185    0.312410    0.000000    0.308225    0.000000    0.000000    0.002092    0.000000    0.000000    0.308225
0.000000    0.001676    0.001676    0.142737    0.074721    0.002514    0.000838    0.000000    0.000000    0.000000    0.000000    0.070531    0.141061    0.493715    0.070531    0.000000
0.000483    0.000000    0.028502    0.114010    0.028986    0.085990    0.029469    0.057005    0.285024    0.256521    0.028502    0.028502    0.028502    0.000000    0.028502    0.000000
0.000000    0.000000    0.000000    0.012121    0.054545    0.042424    0.060606    0.200000    0.145455    0.163636    0.060606    0.072727    0.115152    0.066667    0.006061    0.000000
```

```
0.013596    0.000000    0.000000    0.003399    0.023793    0.007145    0.034753    0.139983    0.173696    0.194298    0.078455    0.089137    0.081646    0.061390    0.051124    0.047586
0.000021    0.000118    0.003084    0.005911    0.011802    0.041235    0.067806    0.109165    0.103584    0.132707    0.159133    0.141461    0.112028    0.058928    0.044181    0.008836
0.007174    0.021521    0.043632    0.024030    0.016414    0.015381    0.043337    0.079205    0.086674    0.100579    0.143916    0.057980    0.129568    0.114778    0.086527    0.029285
0.003741    0.000147    0.011616    0.006790    0.025302    0.029584    0.048641    0.024615    0.101928    0.125369    0.121971    0.183968    0.135957    0.123695    0.024374    0.032302
0.000000    0.074459    0.080229    0.111066    0.052994    0.099873    0.058681    0.066904    0.036809    0.050062    0.058166    0.055412    0.019823    0.097117    0.041646    0.096759
#Number and vector of years of age compositions for recreational fishery
1
2011  #Used to represented years pooled over 2003, 2008, 2009-2011
#Sample size of pooled age comp data (first row avg number observed trips; second row total number observed fish)
4.0   #average number of trips (2003, 2008, 2009-11)
96.0  #total number of observed fish (2003, 2008, 2009-11)
#pooled age composition sample (2003, 2008, 2009-11) from recreational fishery--combined across gear and weighted by number of trips--last age is a plus group (1 to 15+)
0.04828 0.09655 0.15988 0.25106 0.13367 0.03414 0.13143 0.02500 0.04500 0.02500 0.02500 0.02500 0.00000 0.00000 0.00000
###Number and vector of years of age compositions for recreational fishery; used to weight annual predictions before pooling
5
2003  2008  2009  2010  2011
###sample sizes of rec age comps by year for pooling (n.trips)
4.0  1.0  5.0  3.0  7.0
##
####################################Commercial handline fishery#########################################################
#Starting and ending years for HL landings time series
1974
2011
#commercial handline (HL)landings vector (1000 lb whole weight) and assumed CVs (HL landings + HL discards)
33.058  56.554  55.852  30.952  68.882  52.265  83.710  293.649  775.417  339.369  166.585  58.309  112.946  94.632  62.548  66.696  112.086  119.882  125.263  55.052  71.107  65.197  117.077  140.361  65.073
78.784  73.559  89.163  140.684  77.370  42.189  59.080  110.532  57.617  209.737  259.912  136.497  19.822
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    0.05    0.05     0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05
#Number and vector of years of commercial HL length compositions (includes HL discards; unweighted)
21
1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004
#Sample size of commercial HL length comp data (first row number observed trips; second row number observed fish)
49    75    46    37    26    31    40    39    29    41    32    46    24    20    17    34    52    48    33    43    46
404   560   278   232   134   136   396   169   190   339   281   375   209    62   156   342   462   334   121   337    624
#commercial HL length comps (3 cm length bins, weighted
0.000000    0.002800    0.005610    0.023940    0.047370    0.090430    0.126580    0.105180    0.154000    0.114960    0.104700    0.085280    0.050080    0.055500    0.014750    0.018820
0.000000    0.000000    0.000000    0.006530    0.028310    0.054550    0.050200    0.138170    0.113740    0.131270    0.187130    0.113900    0.085480    0.052430    0.027390    0.010890
0.000000    0.000000    0.000000    0.012470    0.016730    0.041670    0.065510    0.081310    0.164230    0.133910    0.157440    0.084950    0.133910    0.062850    0.017840    0.027170
0.000000    0.005360    0.005360    0.045220    0.070500    0.072740    0.098160    0.100440    0.152130    0.085650    0.166620    0.103220    0.043550    0.019100    0.021340    0.010630
0.000000    0.009590    0.057970    0.089300    0.076720    0.049450    0.062020    0.066070    0.131600    0.126060    0.117440    0.125540    0.057970    0.030260    0.000000    0.000000
0.000000    0.005590    0.039120    0.055880    0.055880    0.078240    0.085950    0.101380    0.154600    0.056680    0.096590    0.083290    0.098720    0.046830    0.030070    0.011180
0.000000    0.002070    0.030970    0.094970    0.141410    0.141210    0.111620    0.082720    0.115750    0.105930    0.070000    0.032670    0.033890    0.017380    0.016510    0.002920
0.000000    0.000000    0.000000    0.043100    0.083950    0.187310    0.098250    0.167710    0.115630    0.085400    0.080720    0.061310    0.038820    0.023500    0.000000    0.014300
0.000000    0.011520    0.036450    0.035820    0.067140    0.096570    0.131140    0.123500    0.145760    0.129500    0.098470    0.052160    0.023920    0.021860    0.024550    0.001660
0.000000    0.000000    0.006050    0.029760    0.030080    0.072090    0.132390    0.164350    0.203260    0.106670    0.111940    0.070710    0.039540    0.024030    0.009150    0.000000
0.000000    0.004000    0.020010    0.028020    0.033600    0.080020    0.119210    0.189610    0.168790    0.155140    0.094420    0.038380    0.038410    0.013590    0.003200    0.013590
0.004670    0.011680    0.020820    0.046110    0.060330    0.109390    0.124930    0.159360    0.138740    0.119640    0.052710    0.052300    0.045290    0.013810    0.031280    0.008940
0.000000    0.001780    0.011030    0.075460    0.064050    0.201120    0.152690    0.150160    0.113510    0.065830    0.071530    0.043760    0.032730    0.016360    0.000000    0.000000
0.000000    0.000000    0.009510    0.009510    0.090750    0.100260    0.109770    0.155530    0.121850    0.104630    0.088180    0.131360    0.028540    0.050130    0.000000    0.000000
0.000000    0.004220    0.050580    0.037940    0.088520    0.096950    0.113940    0.136460    0.118150    0.178740    0.084430    0.063350    0.026740    0.000000    0.000000    0.000000
0.000000    0.003990    0.003990    0.037450    0.090860    0.189710    0.157790    0.115870    0.113180    0.059770    0.091370    0.078820    0.019390    0.030950    0.003830    0.003020
0.005110    0.007240    0.009790    0.032810    0.062200    0.121870    0.211760    0.178130    0.131660    0.055810    0.086910    0.054100    0.022580    0.006390    0.009370    0.004260
0.000000    0.008120    0.011120    0.042970    0.077190    0.127590    0.156520    0.137790    0.139850    0.103210    0.103940    0.043040    0.029360    0.017810    0.000000    0.001500
0.000000    0.000840    0.018070    0.027100    0.056390    0.129160    0.137690    0.211300    0.138190    0.092520    0.085330    0.076300    0.018070    0.009030    0.000000    0.000000
0.000000    0.000790    0.003280    0.027410    0.097050    0.188830    0.176050    0.157180    0.107760    0.044530    0.062260    0.072980    0.035640    0.006560    0.013110    0.006560
0.000000    0.006930    0.029390    0.069760    0.063340    0.123390    0.154980    0.156560    0.144380    0.095290    0.056690    0.047270    0.022740    0.007100    0.011090    0.011090
#number and vector of years of age comp data for comm HL fishery
7
2005  2006  2007  2008  2009  2010  2011  #yrs for HL fishery age comps
#Sample size of comm HL age comp data (first row avg number observed trips; second row average number observed fish)
11.0   8.0  30.0   48.0   53.0   68.0   32.0     #HL fishery ntrips
30.0  16.0  87.0  107.0  122.0  180.0  105.0     #HL fishery nfish
#commercial HL age comps (ages 1 to 15+)
0.000000    0.000000    0.019608    0.117647    0.117647    0.196078    0.254902    0.137255    0.039216    0.039216    0.058824    0.000000    0.000000    0.000000    0.019608
0.000000    0.000000    0.000000    0.000000    0.065217    0.108696    0.195652    0.152174    0.173913    0.043478    0.043478    0.043478    0.021739    0.043478    0.108696
0.000000    0.018018    0.036036    0.144144    0.189189    0.144144    0.117117    0.162162    0.045045    0.063063    0.036036    0.018018    0.009009    0.009009    0.009009
0.000000    0.000000    0.035211    0.119718    0.147887    0.281690    0.161972    0.098592    0.063380    0.035211    0.000000    0.014085    0.007042    0.014085    0.023430
0.001567    0.004702    0.018809    0.028213    0.128527    0.300940    0.272727    0.119122    0.064263    0.017241    0.010972    0.004702    0.009404    0.003135    0.014107
0.000000    0.001045    0.028213    0.080460    0.142111    0.226750    0.252874    0.129572    0.073145    0.031348    0.012539    0.005225    0.003135    0.004180    0.014960
0.000000    0.007396    0.050296    0.081361    0.190828    0.208580    0.192308    0.144970    0.053254    0.032544    0.004438    0.011834    0.007396    0.004438    0.010355
####################################Commercial longline fishery#########################################################
#Starting and ending years for LL landings time series
1979
2011
#commercial longline (LL)landings vector (1000 lb whole weight) and assumed CVs (no LL discards available)
5.891  34.461  107.641  406.280  317.818  339.574  333.759  107.255  49.017  43.252  44.450  60.300  70.784  151.578  133.940  112.901  103.386  31.270  76.508  41.413  36.396  34.077  35.881  124.663  33.687
27.003  18.364  47.358  5.529  185.998  198.938  291.214  113.042
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    0.05     0.05    0.05     0.05     0.05     0.05     0.05
#Number and vector of years of commercial LL length compositions (weighted)
22
1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004  2005
#Sample size of commercial LL length comp data (first row number observed trips; second row number observed fish)
17   24   15   9    8   6    9   14    42    73   24   23   13    6   5   9    9   17   28   19   18   7
638  1023  430  95  155  73  315  354  1550  3663  345  372  383  137  123  72  118  400  509  248  290  87
#commercial LL length comps (3 cm length bins, weighted)
0.000000    0.000000    0.000000    0.000000    0.000536    0.002999    0.010818    0.018958    0.033741    0.336569    0.191903    0.157733    0.019922    0.166945    0.029884    0.021101
0.000000    0.000129    0.000000    0.000258    0.005665    0.025921    0.077657    0.134413    0.112735    0.136921    0.189310    0.153332    0.100035    0.043346    0.009921    0.007411
0.000000    0.000000    0.000000    0.000000    0.011313    0.022627    0.029662    0.077447    0.165638    0.216116    0.192533    0.093883    0.067429    0.051364    0.036472    0.019305
0.000000    0.000000    0.000000    0.000000    0.000000    0.042059    0.028226    0.047044    0.107920    0.215278    0.149417    0.153280    0.065300    0.065300    0.023241    0.065300
0.000000    0.000000    0.000000    0.004214    0.004214    0.010759    0.036489    0.042586    0.138964    0.269500    0.197507    0.108122    0.083289    0.042586    0.038372    0.004214
0.000000    0.000000    0.000000    0.000000    0.013699    0.095890    0.041096    0.082192    0.150685    0.123288    0.260274    0.123288    0.054795    0.041096    0.000000    0.000000
0.001804    0.002985    0.002985    0.007774    0.026930    0.038936    0.048647    0.065442    0.094800    0.180437    0.187389    0.155632    0.117357    0.028735    0.003608    0.036537
0.000000    0.000000    0.001516    0.005270    0.019738    0.060278    0.112468    0.112429    0.185596    0.181150    0.154387    0.066207    0.045850    0.025731    0.015810    0.008301
0.000000    0.000000    0.000101    0.000202    0.005113    0.053317    0.078339    0.156384    0.197427    0.184751    0.202182    0.047516    0.033326    0.016516    0.008970    0.006743
0.000000    0.000000    0.000842    0.002662    0.016168    0.052356    0.092304    0.116441    0.182828    0.196588    0.162915    0.076341    0.052038    0.027808    0.012438    0.005144
0.000000    0.001869    0.003739    0.000935    0.027220    0.075117    0.107086    0.110825    0.174725    0.174687    0.180220    0.062814    0.031931    0.022570    0.024417    0.001870
0.000000    0.000000    0.000000    0.014405    0.015490    0.042323    0.161681    0.122952    0.144637    0.239646    0.160934    0.048674    0.023290    0.025968    0.000000    0.000000
0.000000    0.000000    0.000000    0.008623    0.029766    0.090167    0.156637    0.234340    0.164429    0.114900    0.088079    0.055342    0.005278    0.020713    0.018295    0.006716
0.000000    0.010288    0.000000    0.010288    0.020577    0.023139    0.100281    0.161970    0.280266    0.205644    0.079622    0.066813    0.028262    0.012850    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.008130    0.016260    0.065041    0.105691    0.178862    0.211382    0.170732    0.073171    0.073171    0.056911    0.016260
0.000000    0.000000    0.000000    0.008770    0.026310    0.030295    0.017340    0.120648    0.043050    0.310792    0.081983    0.050220    0.004185    0.086168    0.000000    0.146826
0.000000    0.000000    0.000000    0.005983    0.017949    0.047863    0.053846    0.100095    0.194206    0.094112    0.124597    0.088699    0.167618    0.048433    0.000000    0.000000
0.000000    0.000000    0.004625    0.007450    0.012096    0.039053    0.100000    0.116930    0.142427    0.141185    0.112884    0.085907    0.093886    0.087004    0.056551    0.000000
```

```
0.000000   0.000000   0.000000   0.004897   0.012242   0.027472   0.059011   0.130553   0.129434   0.172346   0.141057   0.079927   0.086733   0.056813   0.069054   0.025023
0.000000   0.000000   0.005153   0.046376   0.051529   0.087599   0.119459   0.073083   0.119515   0.133533   0.146221   0.097961   0.058123   0.011747   0.014074   0.022496
0.000000   0.000000   0.002514   0.024982   0.030010   0.094194   0.170427   0.153584   0.124282   0.117981   0.102273   0.079739   0.041377   0.017785   0.010841   0.016655
0.000000   0.000000   0.000000   0.000000   0.033837   0.067674   0.203022   0.135348   0.187395   0.205605   0.051401   0.051401   0.021439   0.005166   0.020147   0.000646
#number and vector of years of age comp data for comm LL fishery
6
2006  2007  2008  2009  2010  2011  #age comps LL fishery
#Sample size of comm LL age comp data (first row avg number observed trips; second row average number observed fish)
8.0   5.0   5.0   48.0   53.0   38.0      #number of trips (2003-11)
30.0  24.0  35.0  516.0  771.0  571.0     #number of fish (2003-11)
#commercial LL age comps (ages 1 to 15+)
0.000000   0.000000   0.000000   0.000000   0.110502   0.133577   0.239777   0.079599   0.145665   0.000000   0.019670   0.030026   0.040931   0.083350   0.116902
0.000000   0.000000   0.000000   0.181940   0.055981   0.295711   0.141762   0.310611   0.000000   0.000000   0.013995   0.000000   0.000000   0.000000   0.000000
0.000000   0.000000   0.050274   0.062402   0.122415   0.412368   0.151134   0.093384   0.067240   0.004776   0.000000   0.000000   0.035946   0.000000   0.000063
0.000528   0.001007   0.006717   0.010554   0.075627   0.356991   0.325130   0.125949   0.059685   0.009598   0.011900   0.000432   0.006334   0.005230   0.004319
0.000000   0.000065   0.006139   0.043509   0.116370   0.266320   0.292866   0.133352   0.086042   0.033131   0.007660   0.004959   0.001478   0.003679   0.004429
0.000000   0.004241   0.019711   0.066420   0.198483   0.249851   0.246625   0.144427   0.035300   0.022638   0.000896   0.005794   0.003345   0.001254   0.001015
########################################Headboat CPUE index#####################################################
#Starting and ending years of HB index
1980
1992
#Observed CPUE and assumed CVs
1.92   1.79   1.20   1.39   0.72   0.67   0.64   0.92   0.70   0.75   0.42   0.67   1.19
0.10   0.16   0.12   0.11   0.16   0.14   0.12   0.13   0.13   0.14   0.16   0.16   0.16
##
########################################Commercial handline CPUE index###########################################################
#Starting and ending years of com handline index
1993
2010
#Observed CPUE and assumed CVs
1.125   0.672   0.638   0.935   0.983   1.163   0.796   1.020   0.910   0.756   0.741   0.875   1.138   1.487   1.182   1.415   0.994   1.169
0.170   0.146   0.103   0.125   0.094   0.101   0.111   0.122   0.123   0.101   0.108   0.100   0.100   0.109   0.094   0.102   0.102   0.107
##
########################################Commercial longline CPUE index###########################################################
#Starting and ending years of com longline index
1993
2004
#Observed CPUE and assumed CVs
2.254   1.024   0.974   0.711   1.530   1.032   0.709   0.501   0.766   1.025   0.891   0.584
0.171   0.177   0.199   0.234   0.145   0.235   0.232   0.202   0.204   0.204   0.205   0.201
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: parameter section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##################Parameter values and initial guesses#############################################################################
####################################################
##prior PDF (1=none, 2=lognormal, 3=normal, 4=beta)
##############################################################
#initial # lower # upper #       #  prior  # prior   # prior #
# guess  # bound # bound # phase #  mean   # var/-CV #  PDF  #
#--------#-------#-------#-------#---------#---------#-------#
############################################################### Biological input ####################################
600.3  300.0  900.0    -3     600.3    -0.5       1     # VonBert Linf (units in mm TL)
0.33   0.05   0.8      -3     0.33     -0.33      1     # VonBert K (units in mm TL)
-0.5   -1.0   0.0      -3     -0.5     -0.28      1     # VonBert t0 (units in mm TL)
0.156  0.05   0.5      -4      0.156   -0.5       1     # CV of length at age
######Below are for separate HL-LL and mrip growth curves using power function function
353.9     200.0    450.0    -3      353.9   -0.5       1     # power fct intercept (units in mm TL) of combined HL-LL fishery
0.255     0.15     0.35     -3      0.255   -0.5       1     # power fct slope (units in mm TL) of combined HL-LL fishery
0.117     0.05     0.5      -4      0.117   -0.5       1     # power fct CV (units in mm TL) of combined HL-LL fishery
##############################################################
256.7     100.0   350.0     -3        256.7   -0.5     1     # power fct intercept (units in mm TL) of mrip fishery
0.380     0.20    0.60      -3        0.380   -0.5     1     # power fct slope (units in mm TL) of mrip fishery
0.113     0.03    0.25       4        0.113   -0.5     1     # power fct CV (units in mm TL) of mrip fishery
##############################################################
0.10
0.05    0.45     -3     0.10     0.092      1     # constant M (used only to compute MSST=(1-M)SSBmsy)
############################################################## SR parameters #####################################
0.836
0.21    0.99       -3     0.836    0.19       4     # SR steepness parameter (0.19 shertzer/conn)
12.0       5.0     18.0      1     12.0      -0.5       1     # SR log_R0 parameter
0.0       -1.0     1.0      -3     0.0       -0.5       1     # SR recruitment autocorrelation (lag 1)
0.6        0.2     1.2       4     0.6       0.15       3     # s.d. of recruitment in log space.
############################################################## Selectivity parameters ##############################
2.5        1.0     10.0      2     2.5       -0.25      3     # recreational age at 50% selectivity:
2.0        0.5     12.0      2     2.0       -0.25      3     # recreational slope of ascending limb
4.25       1.0     10.0      2     4.25      -0.5       3     # commercial handline age at 50% selectivity
2.10       0.5     12.0      2     2.10      -0.5       3     # commercial HL slope of ascending limb
4.25       1.0     10.0      2     4.25      -0.5       3     # commercial longline age at 50% selectivity
2.10       0.5     12.0      2     2.10      -0.5       3     # commercial LLslope of ascending limb
############################################################## Index catchability parameters #######################
-10.0      -14    -4.0       1     -10.0     -0.5       1     # Com LL CPUE (log q)
-10.0      -14    -4.0       1     -10.0     -0.5       1     # HB CPUE (log q)
-10.0      -14    -4.0       1     -10.0     -0.5       1     # Com HL CPUE (log q)
############################################################## Fishing mortality parameters ########################
0.005       0.0   0.01       3      0.005    -0.25      3     # initial F (not log space)
-4.0       -10.0  0.0        1     -4.0      -0.5       1     # recreational log mean F
-4.0       -10.0  0.0        1     -4.0      -0.5       1     # commercial HL log mean F
-4.0       -10.0  0.0        1     -4.0      -0.5       1     # commercial LL log mean F
##### Dev vectors ######################################################################################
#########################
# lower # upper #       #
# bound # bound # phase #
#-------#-------#-------#
-12     12      2    # mrip F devs
-10     10      2    # commercial HL F devs
-10     10      2    # commercial LL F devs
-10     10      3   # recruitment devs
-10     10      3    # initial age structure deviations
##
# mrip F dev initial guesses
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
# commercial HL F dev initial guesses
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
# commercial LL F dev initial guesses
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#initialize single vector of rec devs
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#vector of deviations around initial age structure KC2
0 0 0 0 0 0 0 0 0 0 0 0 0 0
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: likelihood weights section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
###################Likelihood Component Weighting################################################################################
##Weights in objective fcn #commented wgts are from same model only fitting to age comps
1.0      #landings
1.0         #mrip length comps
1.0         #com HL length comps
1.0         #com LL length comps
0.826     #mrip age comps
0.826       #com HL age comps
0.694       #com LL age comps
0.408      #com LL index
0.544       #HB index
0.619       #com HL 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)
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: miscellaneous stuff section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
#length-weight (FL-whole wgt) coefficients a and b, W=aL^b, (W in kg, FL in mm)--sexes combined
6.54E-9
3.11
#weight-gonad weight (whole wgt-gondad weight) coefficients a and b, ln(GW)=a+b*W (units=g)
9.509
0.743
##
#time-invariant vector of fraction mature-at-age for females (ages 1-15+)
0.1  0.25  0.5  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00
##
#time-invariant vector of fraction active spawner at age (ages 1-15+)
1.0   1.0   1.0   1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0
##
#time-invariant vector of proportion female (ages 1-15+)--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  0.5  0.5  0.5
##
#time of year (as fraction) for spawning: end of May
0.42
##
#age-dependent natural mortality at age (ages 1-15+)
0.295  0.199  0.158  0.136  0.124  0.116  0.111  0.107  0.105  0.103  0.102  0.101  0.100  0.100  0.100
##
#Max observed age
43
##
#Spawner-recruit parameters
# SR function switch (integer 1=Beverton-Holt, 2=Ricker)
1
##
#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)
5
#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
##
#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 #com HL
1.0 #com LL
##
#threshold sample sizes (greater than or equal to) for age comps
1.0 #MRFSS
1.0 #com HL
1.0 #com LL
##
#Ageing error matrix (columns are true age 1-15+, rows are ages as read for age comps: columns should sum to one)
1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000     0.000    0.000    0.000
```

```
0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      1.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    1.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    1.000
#USE LANDINGS GROWTH CURVE FLAG USE 0.0 for the population growth rate and 1.0 for the fishery landings growth rate
1.0
##
999 #end of data file flag
##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)
5
#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
##
#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 #com HL
1.0 #com LL
##
#threshold sample sizes (greater than or equal to) for age comps
1.0 #MRFSS
1.0 #com HL
1.0 #com LL
##
#Ageing error matrix (columns are true age 1-15+, rows are ages as read for age comps: columns should sum to one)
1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000    0.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000      0.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      1.000    0.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    1.000    0.000
0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000      0.000    0.000    1.000
#USE LANDINGS GROWTH CURVE FLAG USE 0.0 for the population growth rate and 1.0 for the fishery landings growth rate
1.0
##
999 #end of data file flag
```