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

Sustainable Fisheries Branch, NMFS Beaufort Lab

SEDAR25-RW03

Date Submitted: 23 September 2011

The Beaufort Assessment Model (BAM) with application to black sea bass: 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 (ADMB Foundation 2011), and its structure and equations are detailed in SEDAR-25-RW-03. 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 populations 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 of reef fishes in the U.S. South Atlantic, such as red porgy, tilefish, snowy grouper, gag grouper, greater amberjack, vermilion snapper, Spanish mackerel, red grouper, and red snapper, as well as in previous benchmark (SEDAR-02) and update assessments of black sea bass.

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 $0 - 11^{+}$, where the oldest age class 11^{+} allowed for the accumulation of fish (i.e., plus group).

Initialization Initial (1978) abundance at age was estimated in the model as follows. First, the equilibrium age structure was computed for ages 1–11 based on natural and fishing mortality (F), where F was set equal to the geometric mean fishing mortality from the first three assessment years (1978-1980) scaled by an estimated multiplier (called $F_{\text{init.ratio}}$). Second, lognormal deviations around that equilibrium age structure were estimated. The deviations were lightly penalized, such that the initial abundance of each age could vary from equilibrium if suggested by early age composition data, but remain estimable if data were uninformative. Given the initial abundance of ages 1–11, initial (1978) abundance of age-0 fish was computed using the same methods as recruits in other years.

A normal prior was applied toward the estimation of $F_{\text{init.ratio}}$. The prior had a mean of 1.0 and a CV of 1.0. The large CV defined the prior to be rather loose.

Natural mortality rate The natural mortality rate (M) was assumed constant over time, but decreasing with age. The form of M as a function of age was based on Lorenzen (1996). The Lorenzen (1996) approach inversely relates the natural mortality at age to mean weight at age W_a by the power function $\mathrm{M}_a{=}\alpha W_a^{\beta}$, where α is a scale parameter and β is a shape parameter. Lorenzen (1996) provided point estimates of α and β for oceanic fishes, which were used for this assessment. As in previous SEDAR assessments, the Lorenzen estimates of M_a were rescaled to provide the same fraction of fish surviving from age-1 through the oldest observed age (11 yr) as would occur with constant $M = 0.38$ 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 (total length, TL) was modeled with the von Bertalanffy equation, and weight at age (whole weight, WW) was modeled as a function of total length. Parameters of growth and conversions (TL-WW) were estimated by the DW and were treated as input to the assessment model. The von Bertalanffy parameter estimates from the DW were $L_\infty = 495.9$, $K = 0.177$, and $t_0 = -0.92$. For fitting length composition data, the distribution of size at age was assumed normal with coefficient of variation (CV) estimated

by the assessment model. A constant CV, rather than constant standard deviation, was suggested by the size at age data. For estimating CV of size at age, a normal prior distribution was applied, with mean (0.18) and CV (0.22) provided by the DW.

Sex transition Black sea bass is a protogynous hermaphrodite. Proportion female at age was modeled with a logistic function, estimated by the DW. The age at 50% transition to male was estimated to be 3.83 yr.

Female maturity and fecundity Female maturity was modeled with a logistic function; the age at 50% female maturity was estimated to be ~ 1 yr. Annual egg production by mature females was computed as eggs spawned per batch, a function of body weight, multiplied by the number of batches per year. The number of batches per year was fixed at 31, as recommended by the DW (Danson 2009). Maturity and fecundity parameters were provided by the DW and treated as input to the assessment model.

Spawning stock Spawning stock was modeled as population fecundity of mature females (i.e., total annual egg production) measured at the time of peak spawning. For black sea bass, peak spawning was considered to occur at the end of March.

In cases when reliable estimates of fecundity are unavailable, spawning biomass is commonly used as a proxy for population fecundity. The previous assessment of black sea bass (SEDAR-02) modeled spawning stock as total mature biomass. For protogynous stocks, use of total mature biomass, rather than that of females or males only, has been found to provide more reliable estimates of management quantities over a broad range of conditions (Brooks et al. 2008).

Recruitment Expected recruitment of age-0 fish was predicted from spawning stock using the Beverton–Holt spawner-recruit model. Annual variation in recruitment was assumed to occur with lognormal deviations.

Steepness, h, a key parameter of the Beverton–Holt model, can be difficult to estimate reliably (Conn et al. 2010). Thus, a (beta) prior distribution was applied to steepness. The prior distribution was estimated through meta-analysis on data from 94 stocks of marine demersal fishes (Shertzer and Conn In Press).

The standard deviation of recruitment (σ_R) in log space was estimated using a normal prior distribution, with mean of 0.6 and CV of 0.25, as suggested by the meta-analysis of Mertz and Myers (1996).

Landings The model included time series of landings from five fleets: commercial lines, commercial pots, commercial trawls, headboat, and general recreational. The commercial trawl time series was extended through 1990 (trawling was banned in January, 1989 within federal waters of the SAFMC's jurisdiction).

Landings were modeled with the Baranov catch equation (Baranov 1918) and were fitted in units of weight (1000 lb whole weight). The DW provided observed landings back to the first assessment year (1978) for each fleet except general recreational, because the MRFSS started in 1981. Thus for years 1978–1980, general recreational landings were predicted in the assessment model (but not fitted to data), by applying the geometric mean recreational F from the years 1981–1983.

Discards As with landings, discard mortalities (in units of 1000 fish) were modeled with the Baranov catch equation (Baranov 1918), which required estimates of discard selectivities and release mortality probabilities. Discards were assumed to have gear-specific mortality probabilities, as suggested by the DW (lines, 0.07; pots with 1.5-inch panels, 0.05; and pots with 2-inch panels, 0.01). Annual discard mortalities, as fitted by the model, were computed by multiplying total discards (tabulated in the DW report) by the gear-specific release mortality probability.

For the commercial fleets, discards from handline and pot gears were combined, and were modeled starting in 1984 with implementation of the 8-inch size limit. Commercial discards prior to 1984 were considered negligible and not modeled. Data on commercial discards were available from the DW starting in 1993. Thus for years 1984–1992, commercial discards were predicted in the assessment model (but not fitted to data), by applying the geometric mean commercial discard F from the years 1993–1998 (the 10-inch limit began in 1999).

For headboat and general recreational fleets, discard time series were assumed to begin in 1978, as observations from MRFSS indicated the occurrence of recreational discards prior to implementation of the 8-inch size limit. Headboat

discard estimates were separated from MRFSS in 1986, and were combined for 1978–1985. Because MRFSS began in 1981, the 1978–1980 general recreational (plus headboat) discards were predicted in the assessment model (but not fitted to data), by applying the geometric mean recreational discard F from the years 1981–1983.

For fishery discard length composition data collected under a size limit regulation, the normal distribution of size at age was truncated at the size limit, such that length compositions of discards would include only fish of sublegal size. Mean length at age of discards were computed from these truncated distributions, and thus average weight at age of discards would differ from those in the population at large. A portion of commercial discards in 2009–2010 consisted of fish that were of legal size as a result of the closed seasons.

Fishing For each time series of landings and discard mortalities, 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 compute as the maximum of F at age summed across fleets.

Selectivities Selectivity curves applied to landings and MARMAP survey gears 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. Selectivities of landings from all fleets were modeled as flat-topped, using a two-parameter logistic function. Selectivities of fishery-dependent indices were the same as those of the relevant fleet.

Selectivity of each fleet was fixed within each block of size-limit regulations, but was permitted to vary among blocks where possible or reasonable. Commercial fisheries experienced three blocks of size-limit regulations: no limit prior to 1983, 8-inch limit during 1983–1999, and 10-inch limit during 1999–2010. Recreational fisheries experienced four blocks of size-limit regulations, which were the same as those of the commercial fisheries but with a 12-inch size limit implemented in 2007.

Age and length composition data are critical for estimating selectivity parameters, and ideally, a model would have sufficient composition data from each fleet over time to estimate distinct selectivities in each period of regulations. That was not the case here, and thus additional assumptions were applied to define selectivities, as follows. Because no age and very few length composition data were available from commercial trawls, selectivity of this fleet was assumed to mirror that of the commercial pots. With no composition data from commercial fleets prior to regulations, commercial line selectivities in the first and second regulatory blocks were set equal, as were commercial pot selectivities, consistent with the DW recommendation that the 8-inch size limit had little effect on commercial fishing. Length composition data from MRFSS were quite noisy, and thus selectivities of recreational headboat and general recreational fleets mirrored each other.

Selectivities of discards were assumed to be dome-shaped. They were partially estimated, assuming that discards consisted primarily of undersized fish, as implied by observed length compositions of discards. The general approach taken was that age-specific values for ages 0–2 were estimated, age 3 was assumed to have full selection, and selectivity for each age 4^+ was set equal to the age-specific probability of being below the size limit, given the estimated normal distribution of size at age. In this way, the descending limb of discard selectivities would change with modification of the size limit. The exception to the above approach was for commercial discards in years 2009–2010, when a commercial quota was in place. For those years, commercial discard selectivity included fish larger than the 10-inch size limit that would have been released during the closed season. The commercial discard selectivity for these years was computed as the combined selectivities of sublegal-sized fish and landed fish from commercial lines and pots, weighted by the geometric mean (2009-2010) of fleet-specific observed discards or landings.

Diffuse priors were used for estimating parameters of selectivity functions. These priors assumed normal distributions with CV = 1.0 and were intended to provide only weak 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 diffuse 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 depending on values of other parameters. When this happens, the likelihood gradient with respect to the aimless parameter approaches zero even if the parameter is not at its globally best value. Diffuse priors help avoid this situation.

Indices of abundance The model was fit to two fishery-independent indices of relative abundance (MARMAP blackfish/snapper traps 1981–1987; and MARMAP chevron traps 1990–2010) and three fishery-dependent indices (headboat 1979–2010; headboat discards 2005–2010; and commercial lines 1993–2010). Predicted indices were conditional on selectivity of the corresponding fleet or survey and were computed from abundance or biomass (as appropriate) at the midpoint of the year. The headboat discard index, although relatively short in duration, tracks young fish and was included as a measure of recruitment strength at the end of the assessment period. All indices were positively correlated, and in most cases, significantly.

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, following SEDAR-02. 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).

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 population fecundity 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 (including discard mortalities) estimated as the full F averaged over the last two years of the assessment. The last two years, rather than three (SEDAR custom), was applied because of the implementation of commercial seasonal closures starting in 2009.

Fitting criterion The fitting criterion was a penalized likelihood approach in which observed landings and discards were fit closely, and observed composition data and abundance indices were fit to the degree that they were compatible. Landings, discards, and index data were fitted using lognormal likelihoods. Length and age composition data were fitted using 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 black sea bass, CVs of landings and discards (in arithmetic space) were assumed equal to 0.05, to achieve a close fit to these time series yet allow some imprecision. In practice, the small CVs are a matter of computational convenience, as they help achieve the desired result of close fits to the landings, while avoiding having to solve the Baranov equation iteratively (which is complex when there are multiple fisheries). Weights on other data components (indices, age/length compositions) were adjusted iteratively, starting from initial weights as follows. The CVs of indices were set equal to the values estimated by the DW. Effective sample sizes of the multinomial components were assumed equal to the number of trips sampled annually, rather than the number of fish measured, reflecting the belief that the basic sampling unit occurs at the level of trip. These initial weights were then adjusted until standard deviations of normalized residuals were near 1.0 (SEDAR24-RW03, SEDAR25-RW05). Weights on four indices (all but the headboat discard index) were then adjusted upward to a value of 2.5 (SEDAR25-RW05), in accordance with the principle that abundance data should be given primacy (Francis 2011), which would seem particularly true when indices are highly correlated.

In addition, a lognormal likelihood was applied to the spawner-recruit relationship. The compound objective function also included several penalties or prior distributions (e.g., on estimated parameters of selectivity functions). Penalties

or priors were applied to maintain parameter estimates near reasonable values, and to prevent the optimization routine from drifting into parameter space with negligible gradient in the likelihood.

Model testing Experiments with a reduced model structure indicated that parameters estimated from the BAM were unbiased and could be recovered from simulated data. Further, the general model structure has been through multiple SEDAR reviews. As an additional measure of quality control, black sea bass 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 black sea bass stock dynamics.

References

ADMB Foundation, 2011. AD Model Builder: automatic differentiation model builder. Available: http://www.admb-project.org.

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

Brooks, E. N., K. W. Shertzer, T. Gedamke, and D. S. Vaughan. 2008. Stock assessment of protogynous fish: evaluating measures of spawning biomass used to estimate biological reference points. Fishery Bulletin **106**:12–23.

Conn, P. B., E. H. Williams, and K. W. Shertzer. 2010. When can we reliably estimate the productivity of fish stocks? Canadian Journal of Fisheries and Aquatic Sciences **67**:511–523.

Danson, B. L., 2009. Estimating reef fish reproductive productivity on artificial and natural reefs off the Atlantic coast of the southeastern United States. Master's thesis, The College of Charleston.

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 age data. Canadian Journal of Fisheries and Aquatic Sciences **39**:1195–1207.

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., and P. B. Conn. In Press. Spawner-recruit relationships of demersal marine fishes: Prior distribution of steepness. Bulletin of Marine Science .

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 black sea bass. 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	$y \in \{1978 \dots 2010\}$
Index of ages	a	$a \in \{0 \dots A\}$, where $A = 11^{+}$
Index of size-limit periods	r	$r \in \{1 \dots 4\}$ where $1 = 1978 - 1983$ (no size limit rec or comm), $2 = 1984 - 1998$ (8-inch limit rec and comm), $3 = 1999 - 2006$ (10-inch limit rec and comm), and $4 = 2007 - 2010$ (12-inch limit rec, 10-inch limit comm); 'rec' = recreational, 'comm' = commercial
Index of length bins	l	$l \in \{1 \dots 41\}$
Length bins	l'	$l' \in \{100, 110, \dots, 600\text{mm}\}$, with midpoint of 10mm bin used to match length compositions. Largest 10 length bins (TL > 500 mm) treated as a plus group, but retained for weight calculations.
Index of fisheries	f	$f \in \{1 \dots 5\}$ where 1 = commercial lines, 2 = commercial pots, 3 = commercial trawls, 4 = recreational headboat, 5 = general recreational
Index of discards	d	$d \in \{1 \dots 3\}$ where 1 = commercial lines and pots (combined), 2 = headboat, 3 = general recreational
Index of CPUE	u	$u \in \{1 \dots 5\}$ where 1 = MARMAP blackfish/snapper traps, 2 = MARMAP chevron traps, 3 = commercial lines, 4 = headboat, 5 = headboat discards
Input Data		
Observed length compositions	$p^{\lambda}_{(f,d,u),l,y}$	Proportional contribution of length bin l in year y to fishery f, d (landings or discards) 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
Length comp. sample sizes	$n^{\lambda}_{(f,d,u),y}$	Effective number of length samples collected in year y from fishery f, discards d, 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.
CVs of landings	$c^{L}_{f,y}$	Assumed 0.05 in arithmetic space
Observed abundance indices	$U_{u,y}$	$u = 1$, commercial lines (weight), $y \in \{1993 \dots 2010\}$ $u = 2$, headboat (weight), $y \in \{1979 \dots 2010\}$ $u = 3$, headboat discards (numbers), $y \in \{2005 \dots 2010\}$ $u = 4$, MARMAP chevron trap (numbers), $y \in \{1990 \dots 2010\}$ $u = 5$, MARMAP blackfish/snapper trap (numbers), $y \in \{1981 \dots 1987\}$
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1 \dots 5\}$ as above. Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean
Observed total discards	$D'_{d,y}$	Discards (1000 fish) in year y from fishery d.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Discard mortality rate	δ_d	Proportion discards by fishery d that die. The DW recommended $\delta_d = 0.07$ for lines, $\delta_d = 0.05$ for 1.5 inch panel pots, and $\delta_d = 0.01$ for 2 inch panel pots.
Observed discard mortalities	$D_{d,y}$	$D_{d,y} = \delta_d D'_{d,y}$
CVs of dead discards	$c^D_{d,y}$	Assumed 0.05 in arithmetic space
Population Model		
Mean length at age	l_a	Total length (midyear); $l_a = L_\infty(1 - \exp[-K(a - t_0 + 0.5)])$ where K, L_∞, and t_0 are parameters estimated by the DW
CV of l_a	$\widehat{c}^\lambda_a$	Estimated coefficient of variation of growth, assumed constant across ages
SD of l_a	σ^λ_a	Standard deviation of growth, $\sigma^\lambda_a = \widehat{c}^\lambda_a l_a$
Age–length conversion of population	$\psi^u_{a,l}$	$\psi^u_{a,l} = \frac{1}{\sqrt{2\pi}(\sigma^\lambda_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\sigma^\lambda_a)^2\right)}$, 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 and is used only to match length comps of fishery independent indices.
Age–length conversion of landings	$\psi^L_{f,a,l,y}$	$\psi^L_{f,a,l,y} = \psi^u_{a,l}$
Age–length conversion of discards	$\psi^D_{d,a,l,y}$	$\psi^D_{d,a,l,y} = \begin{cases} \frac{1}{\sqrt{2\pi}(\sigma^\lambda_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\sigma^\lambda_a)^2\right)} & : l_a < l_{\text{limit}} \\ 0 & : \text{otherwise} \end{cases}$ where l_{limit} is the size limit for fishery d in year y (and could be treated as ∞ prior to regulations). Annual matrices $\psi^D_{d,\cdot,\cdot,y}$ are rescaled to sum to one within ages, with the largest size a plus group.
Mean length at age of landings and discards	$\xi^{L,D}_{(f,d),a,y}$	Mean length at age from $\psi^L_{f,a,y}$ for landings or $\psi^D_{d,a,y}$ for discards
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 and discards	$w^{L,D}_{(f,d),a,y}$	Computed from length at age by $w^{L,D}_{(f,d),a,y} = \theta_1(\xi^{L,D}_{(f,d),a,y})^{\theta_2}$
Natural mortality rate	M_a	Function of weight at age (w_a): $M_a = \alpha w_a^\beta$, with estimates of α and β from Lorenzen (1996). Lorenzen M_a then rescaled based on Hoenig estimate.
Proportion male at age	ρ_a	Logistic increase with age; assumed constant across years
Proportion female at age	$1 - \rho_a$	Complement of above
Proportion females mature at age	m_a	Logistic increase with age. All males assumed mature.
Batch fecundity at age	E_a	Eggs spawned per batch, $\log(E_a) = \theta_3 + \theta_4 w_a$ where θ_3 and θ_4 are parameters from the DW
Number annual batches at age	b_a	Number of batches spawned per yr; assumed constant (31) across ages
Annual fecundity at age	$\mathcal{F}_a$	$\mathcal{F}_a = b_a E_a$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Spawning date	t_{spawn}	Fraction denoting the proportional time of year when spawning occurs. Set to 0.25, assuming peak spawning occurs at end of March
Fishery and index selectivities	$s_{(f,u),a,r}$	$s_{(f,u),a,r} = \frac{1}{1+\exp\left[-\widehat{\eta}_{(f,u),r}\left(a-\widehat{\alpha}_{(f,u),r}\right)\right]}$ where $\widehat{\eta}_{(f,u),r}$ and $\widehat{\alpha}_{(f,u),r}$ are estimated parameters. Not all parameters were estimated for each fishery (or index) and each period of regulations; some parameters were fixed as described in the text. For instance, the selectivity of commercial trawls was assumed equal to that of commercial pots. Commercial line selectivity was set equal between the first and second regulatory periods, as was commercial pot selectivity. Selectivity of headboat and general recreational fleets were set equal.
Discard selectivities	$s'_{d,a,r}$	$s'_{d,0,r} = \text{logit}^{-1}(\widehat{s'_0})$, $s'_{d,1,r} = \text{logit}^{-1}(\widehat{s'_1})$, and $s'_{d,2,r} = \text{logit}^{-1}(\widehat{s'_2})$, where $\widehat{s'_a}$ are estimated parameters; $s'_{d,3,r} = 1.0$; $s'_{d,4^+,r}$ set equal to the age-specific probability of total length below the sector-specific size limit in period r. The exception was commercial discard selectivity during the closure, $s'_{1,a,2009-2010}$, modeled as the weighted average of $s'_{1,a,4}$, $s_{1,a,4}$, and $s_{2,a,4}$ (see text).
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 and $s_{f,a,y} = s_{f,a,r}$ for y in the years represented by r
Fishing mortality rate of discards	$F^D_{d,a,y}$	$F^D_{d,a,y} = s'_{d,a,r}\widehat{F}^D_{d,y}$ where $\widehat{F}^D_{d,y}$ is an estimated fully selected fishing mortality rate of discards by fishery
Total fishing mortality rate	$F_{a,y}$	$F_{a,y} = \sum_f F_{f,a,y} + \sum_d F^D_{d,a,y}$
Total mortality rate	$Z_{a,y}$	$Z_{a,y} = M_a + F_{a,y}$
Apical F	F_y	$F_y = \max(F_{a,y})$
Abundance at age	$N_{a,y}$	$N_{0,1978} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}\phi_{init}-0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)\phi_{init}}$ $\widehat{N}_{1+,1978}$ equilibrium conditions expected given assumptions about initial fishing mortality (described below) $N_{0,y} = \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_y)$ for $y > 1978$ $N_{a+1,y} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (0 \dots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ Parameters $\widehat{R}_0$ (asymptotic maximum recruitment) and $\widehat{h}$ (steepness) are estimated parameters 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. 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 late March to correspond with peak spawning $N''_{a,y} = \exp(-t_{\text{spawn}}Z_{a,y})N_{a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
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	$\phi_0 = \sum_{a=0}^{A} NPR_a \rho_a m_a \mathcal{F}_a$ In units of population fecundity (number eggs) of mature females.
Initial spawning biomass per recruit	ϕ_{init}	$\phi_{init} = \sum_{a=0}^{A} NPR_a^{init} \rho_a m_a \mathcal{F}_a$ In units of population fecundity (number eggs) of mature females.
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y} \rho_a m_a \mathcal{F}_a$ in units of fecundity (egg number) of mature females.
Initialization mortality at age	Z_a^{init}	$Z_a^{init} = M_a + F^{init}$ where F^{init} is an estimated initialization F, assumed to be the geometric mean fishing mortality from the first three assessment years (1978-1980) scaled by an estimated multiplier $\widehat{F}_{\text{init.ratio}}$
Initial equilibrium abundance at age	N_a^{eq}	Equilibrium age structure given Z_a^{init}
Population biomass	B_y	$B_y = \sum_a N_{a,y} w_a$
Landing at age in numbers	$L'_{f,a,y}$	$L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}} N_{a,y}[1 - \exp(-Z_{a,y})]$
Landing at age in weight	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y} L'_{f,a,y}$
Discard mortalities at age in numbers	$D'_{d,a,y}$	$D'_{d,a,y} = \frac{F^D_{d,a,y}}{Z_{a,y}} N_{a,y}[1 - \exp(-Z_{a,y})]$
Discard mortalities at age in weight	$D''_{d,a,y}$	$D''_{d,a,y} = w^D_{d,a,y} D'_{d,a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Index catchability	$q_{u,y}$	$q_{u,1978} = \widehat{q}_u^0 f(\text{density})$ $q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ for $y \geq 1978$ 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 1978, $f_y(\text{trend})$ follows, $f_y(\text{trend}) = \begin{cases} 1.0 & :y = 1978 \\ f_{y-1}(\text{trend}) * (\text{y} - 1978)\beta_{\text{q}} & :1978 < y \leq 2003 \\ f_{2003}(\text{trend}) & :2003 < y \end{cases}$ 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"). For this assessment, catchability was assumed constant in the base run and the catchability function $f_y(\text{trend})$ was applied as a sensitivity run as described in the SEDAR-25 black sea bass assessment. Density dependence and random walks were not applied in the base run.
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \sum_a L''_{f,a,y}$
Predicted discard mortalities	$\breve{D}_{d,y}$	$\breve{D}_{d,y} = \sum_a D'_{d,a,y}$
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 length compositions of discards	$\breve{p}^{\lambda}_{d,l,y}$	$\breve{p}^{\lambda}_{d,l,y} = \frac{\sum_a \psi^D_{d,a,l,y} D'_{d,a,y}}{\sum_a D'_{d,a,y}}$
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	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \begin{cases} \widehat{q}_{u,y} \sum_a w^L_{u,a,y} N'_{a,y} s_{u,a,r} & : \quad u = 3,4 \\ \widehat{q}_{u,y} \sum_a N'_{a,y} s_{u,a,r} & : \quad u = 1,2,5 \end{cases}$ where $s_{u,a,r}$ is the selectivity of the relevant fishery in the year corresponding to y.

Table 2.1. (continued)

Quantity	Symbol	Description or definition
Objective Function		
Multinomial length compositions	Λ_1	$\Lambda_1 = -\sum_{f,d,u}\sum_{y}\left[\omega^{\lambda}_{(f,d,u)} n^{\lambda}_{(f,d,u),y} \sum_{l}(p^{\lambda}_{(f,d,u),l,y}+x)\log\left(\frac{(\breve{p}^{\lambda}_{(f,d,u),l,y}+x)}{(p^{\lambda}_{(f,d,u),l,y}+x)}\right)\right]$ where $\omega^{\lambda}_{(f,d,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term. Bins are 10 mm wide.
Multinomial age compositions	Λ_2	$\Lambda_2 = -\sum_{f,u}\sum_{y}\left[\omega^{\alpha}_{(f,u)} n^{\alpha}_{(f,u),y} \sum_{a}(p^{\alpha}_{(f,u),a,y}+x)\log\left(\frac{(\breve{p}^{\alpha}_{(f,u),a,y}+x)}{(p^{\alpha}_{(f,u),a,y}+x)}\right)\right]$ where $\omega^{\alpha}_{(f,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term.
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 discard mortalities	Λ_4	$\Lambda_4 = \sum_{d}\sum_{y}\frac{\left[\log\left((\delta_d D_{d,y}+x)/(\breve{D}_{d,y}+x)\right)\right]^2}{2(\sigma^D_{d,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^D_{d,y} = \sqrt{\log(1+(c^D_{d,y}/\omega^D_d)^2)}$, with $\omega^D_d = 1$ a preset weight.
Lognormal CPUE	Λ_5	$\Lambda_5 = \sum_{u}\sum_{y}\frac{\left[\log\left((U_{u,y}+x)/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma^U_{u,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^U_{u,y} = \sqrt{\log(1+(c^U_{u,y}/\omega^U_u)^2)}$, with ω^U_u a preset weight (selected by iterative re-weighting).
Lognormal recruitment deviations	Λ_6	$\Lambda_6 = \omega_6\left[\frac{[R_{1978}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y>1978}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where R_y are recruitment deviations in log space, n is the number of years, $\omega_6 = 1$ 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-25 base run).
Additional constraint on early recruitment deviation	Λ_7	$\Lambda_7 = \omega_7\left[\frac{[R_{1978}+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y=1979}^{Y_1}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where Y_1 is the last year to apply this additional penalty and ω_7 is a preset weight, with ω_7=0.0 for the SEDAR-25 black sea bass base run.
Additional constraint on final recruitment deviations	Λ_8	$\Lambda_8 = \omega_8\left[\sum_{y=Y_2}^{Y}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where Y_2 is the first year to apply this additional penalty, Y is the terminal year, and ω_8 is a preset weight, with ω_8=0.0 for the SEDAR-25 black sea bass base run.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Penalty on random walk on catchability	Λ_9	$\Lambda_9 = \omega_9 \sum_u \sum_y \frac{\epsilon^2_{u,y}}{2(\sigma^q_u)^2}$ where ω_9 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-25 black sea bass base run, thus ω_9=0.0.
Penalty on initial age structure	Λ_{10}	$\Lambda_{10} = \sum_{a=1}^{A} (\widehat{N}_{a,1978} - N^{eq}_a)^2$ where N^{eq}_a is the equilibrium age structure given the initial F, as defined previously.
Prior distributions	Λ_{11}	Λ_{11} is the sum of penalty terms used to implement prior distributions on several parameters. A beta prior was applied to $\widehat{h}$, and normal priors were applied to $\widehat{\sigma}_R$, $\widehat{c^{\lambda}_a}$, $\widehat{F}_{\text{init.ratio}}$, $\widehat{\eta}_{(f,u),r}$, $\widehat{\alpha}_{(f,u),r}$, and $\widehat{s'}_{0-2}$. Normal distributions required a value to describe variance. Empirical estimates of standard deviation were available and therefore used for c^{λ}_a and σ_R. All other normal priors assumed CV=1 (i.e., diffuse priors).
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

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR25 Assessment: Black Sea Bass, June 2011
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

DATA_SECTION

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

// Starting and ending year of the model (year data starts)
init_int styr;
init_int endyr;
//!!cout << styr << endl;

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

//Reg blocks -- 3 possible periods of comm size regs: styr-83 no restrictions, 1984-98 8-inch TL, 1999-2010 10-in TL
//           -- 4 possible periods of recr size regs: styr-83 no restrictions, 1984-98 8-inch TL, 1999-2006 10-in TL, 2007-2010 12-in TL
init_int endyr_period1;
init_int endyr_period2;
init_int endyr_recr_period3;
//first yr commercial fisheries were closed due to quotas
init_int styr_comm_closed;
//size limits
init_number limit_8in;   //8 inch limit in mm
init_number limit_10in;  //10 inch limit in mm
init_number limit_12in;  //12 inch limit in mm
init_number limit_disc;  //max size applied to discards in block one, prior to fed regs

//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-styr_rec_dev+1.;
 END_CALCS

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

//Vector of lengths for length bins (mm)(midpoint) and bins used in computation of plus group
init_ivector lenbins(1,nlenbins);
init_ivector lenbins_plus(1,nlenbins_plus);
int nlenbins_all;    //largest size class used to compute average lengths and weights

//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   nlenbins_all=nlenbins+nlenbins_plus;
 END_CALCS


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

//#########################################################################
//###MARMAP bft/fst#################
//CPUE
init_int styr_Mbft_cpue;
init_int endyr_Mbft_cpue;
init_vector obs_Mbft_cpue(styr_Mbft_cpue,endyr_Mbft_cpue);   //Observed CPUE
init_vector Mbft_cpue_cv(styr_Mbft_cpue,endyr_Mbft_cpue);    //CV of cpue

// Length Compositions (1 cm bins)
init_int nyr_Mbft_lenc;
init_ivector yrs_Mbft_lenc(1,nyr_Mbft_lenc);
init_vector nsamp_Mbft_lenc(1,nyr_Mbft_lenc);
init_vector nfish_Mbft_lenc(1,nyr_Mbft_lenc);
init_matrix obs_Mbft_lenc(1,nyr_Mbft_lenc,1,nlenbins);

// Age Compositions
init_int nyr_Mbft_agec;
```

```
init_ivector yrs_Mbft_agec(1,nyr_Mbft_agec);
init_vector nsamp_Mbft_agec(1,nyr_Mbft_agec);
init_vector nfish_Mbft_agec(1,nyr_Mbft_agec);
init_matrix obs_Mbft_agec(1,nyr_Mbft_agec,1,nages);

//#########################################################################
//###MARMAP cvt#################
//CPUE
init_int styr_Mcvt_cpue;
init_int endyr_Mcvt_cpue;
init_vector obs_Mcvt_cpue(styr_Mcvt_cpue,endyr_Mcvt_cpue);   //Observed CPUE
init_vector Mcvt_cpue_cv(styr_Mcvt_cpue,endyr_Mcvt_cpue);    //CV of cpue

// Age Compositions
init_int nyr_Mcvt_agec;
init_ivector yrs_Mcvt_agec(1,nyr_Mcvt_agec);
init_vector nsamp_Mcvt_agec(1,nyr_Mcvt_agec);
init_vector nfish_Mcvt_agec(1,nyr_Mcvt_agec);
init_matrix obs_Mcvt_agec(1,nyr_Mcvt_agec,1,nages);

//#########################################################################
//###################Commercial Hook and Line fishery ########################
//CPUE
init_int styr_cL_cpue;
init_int endyr_cL_cpue;
init_vector obs_cL_cpue(styr_cL_cpue,endyr_cL_cpue);//Observed CPUE
init_vector cL_cpue_cv(styr_cL_cpue,endyr_cL_cpue); //CV of cpue

// Landings  (1000 lb whole weight)
init_int styr_cL_L;
init_int endyr_cL_L;
init_vector obs_cL_L(styr_cL_L,endyr_cL_L);   //vector of observed landings by year
init_vector cL_L_cv(styr_cL_L,endyr_cL_L);    //vector of CV of landings by year

// Discards (1000 fish)
init_int styr_cL_D;
init_int endyr_cL_D;
init_vector obs_cL_released(styr_cL_D,endyr_cL_D); //vector of observed releases by year, gets multiplied by discard mortality for fitting
init_vector cL_D_cv(styr_cL_D,endyr_cL_D);         //vector of CV of discards by year
// Discards (1000 fish) during closed season
init_int styr_cL_closed_D;
init_int endyr_cL_closed_D;
init_vector obs_cL_closed_released(styr_cL_closed_D,endyr_cL_closed_D); //vector of observed releases by year, gets multiplied by discard mortality for fitting

// Length Compositions (1 cm bins)
init_int nyr_cL_lenc;
init_ivector yrs_cL_lenc(1,nyr_cL_lenc);
init_vector nsamp_cL_lenc(1,nyr_cL_lenc);
init_vector nfish_cL_lenc(1,nyr_cL_lenc);
init_matrix obs_cL_lenc(1,nyr_cL_lenc,1,nlenbins);
// Age Compositions
init_int nyr_cL_agec;
init_ivector yrs_cL_agec(1,nyr_cL_agec);
init_vector nsamp_cL_agec(1,nyr_cL_agec);
init_vector nfish_cL_agec(1,nyr_cL_agec);
init_matrix obs_cL_agec(1,nyr_cL_agec,1,nages);


//#########################################################################
//##Commercial pot (+ other) fleet
// Landings (1000 lb whole weight)
init_int styr_cP_L;
init_int endyr_cP_L;
init_vector obs_cP_L(styr_cP_L,endyr_cP_L);
init_vector cP_L_cv(styr_cP_L,endyr_cP_L);    //vector of CV of landings by year

// Discards (1000 fish)
init_int styr_cP_D;
init_int endyr_cP_D;
init_vector obs_cP_released(styr_cP_D,endyr_cP_D); //vector of observed releases by year, gets multiplied by discard mortality for fitting
init_vector cP_D_cv(styr_cP_D,endyr_cP_D);         //vector of CV of discards by year
// Discards (1000 fish) during closed season
init_int styr_cP_closed_D;
init_int endyr_cP_closed_D;
init_vector obs_cP_closed_released(styr_cP_closed_D,endyr_cP_closed_D); //vector of observed releases by year, gets multiplied by discard mortality for fitting

// Length Compositions (1 cm bins)
init_int nyr_cP_lenc;
init_ivector yrs_cP_lenc(1,nyr_cP_lenc);
init_vector nsamp_cP_lenc(1,nyr_cP_lenc);
init_vector nfish_cP_lenc(1,nyr_cP_lenc);
init_matrix obs_cP_lenc(1,nyr_cP_lenc,1,nlenbins);
init_int nyr_cP_lenc_pool;     //years and weights to pool predicted cP length comps to match pooled observations
init_ivector yrs_cP_lenc_pool(1,nyr_cP_lenc_pool);
init_vector nsamp_cP_lenc_pool(1,nyr_cP_lenc_pool);

// Age Compositions
init_int nyr_cP_agec;
init_ivector yrs_cP_agec(1,nyr_cP_agec);
init_vector nsamp_cP_agec(1,nyr_cP_agec);
init_vector nfish_cP_agec(1,nyr_cP_agec);
init_matrix obs_cP_agec(1,nyr_cP_agec,1,nages);

//#########################################################################
//#########################################################################
//##Commercial Trawl fleet
// Landings (1000 lb whole weight)
init_int styr_cT_L;
init_int endyr_cT_L;
init_vector obs_cT_L(styr_cT_L,endyr_cT_L);
init_vector cT_L_cv(styr_cT_L,endyr_cT_L);    //vector of CV of landings by year


//#########################################################################
```

```
//##############################Headboat fleet ######################################
//CPUE
init_int styr_HB_cpue;
init_int endyr_HB_cpue;
init_vector obs_HB_cpue(styr_HB_cpue,endyr_HB_cpue);//Observed CPUE
init_vector HB_cpue_cv(styr_HB_cpue,endyr_HB_cpue); //CV of cpue
//###HBD index (headboat discards from at sea observer program#################
init_int styr_HBD_cpue;
init_int endyr_HBD_cpue;
init_vector obs_HBD_cpue(styr_HBD_cpue,endyr_HBD_cpue);   //Observed CPUE
init_vector HBD_cpue_cv(styr_HBD_cpue,endyr_HBD_cpue);    //CV of cpue
// Landings (1000 lb)
init_int styr_HB_L;
init_int endyr_HB_L;
init_vector obs_HB_L(styr_HB_L,endyr_HB_L);
init_vector HB_L_cv(styr_HB_L,endyr_HB_L);
// Discards (1000s)
init_int styr_HB_D;
init_int endyr_HB_D;
init_vector obs_HB_released(styr_HB_D,endyr_HB_D);  //vector of observed releases by year, multiplied by discard mortality for fitting
init_vector HB_D_cv(styr_HB_D,endyr_HB_D);          //vector of CV of discards by year
// Length Compositions (1 cm bins) of landings
init_int nyr_HB_lenc;
init_ivector yrs_HB_lenc(1,nyr_HB_lenc);
init_vector nsamp_HB_lenc(1,nyr_HB_lenc);
init_vector nfish_HB_lenc(1,nyr_HB_lenc);
init_matrix obs_HB_lenc(1,nyr_HB_lenc,1,nlenbins);
// Age compositions of landings
init_int nyr_HB_agec;
init_ivector yrs_HB_agec(1,nyr_HB_agec);
init_vector nsamp_HB_agec(1,nyr_HB_agec);
init_vector nfish_HB_agec(1,nyr_HB_agec);
init_matrix obs_HB_agec(1,nyr_HB_agec,1,nages);
//Length Compositions (1 cm bins) of HB discards
init_int nyr_HB_D_lenc;
init_ivector yrs_HB_D_lenc(1,nyr_HB_D_lenc);
init_vector nsamp_HB_D_lenc(1,nyr_HB_D_lenc);
init_vector nfish_HB_D_lenc(1,nyr_HB_D_lenc);
init_matrix obs_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);


//#########################################################################
//############################mrip recreational fleet ##############################
// Landings (1000 lb)
init_int styr_mrip_L;
init_int endyr_mrip_L;
init_vector obs_mrip_L(styr_mrip_L,endyr_mrip_L);
init_vector mrip_L_cv(styr_mrip_L,endyr_mrip_L);
// Discards (1000s)
init_int styr_mrip_D;
init_int endyr_mrip_D;
init_vector obs_mrip_released(styr_mrip_D,endyr_mrip_D); //vector of observed releases by year, multiplied by discard mortality for fitting
init_vector mrip_D_cv(styr_mrip_D,endyr_mrip_D);         //vector of CV of discards by year
// Length Compositions (1 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);
//init_int nyr_mrip_lenc_pool;     //years and weights to pool predicted mrip length comps to match pooled observations
//init_ivector yrs_mrip_lenc_pool(1,nyr_mrip_lenc_pool);
//init_vector nsamp_mrip_lenc_pool(1,nyr_mrip_lenc_pool);
// 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);

//#################Parameter values and initial guesses ################################
//#########################################################################
//Discard mortality constants
init_number set_Dmort_HL;   //handline (commercial and recreational)
init_number set_Dmort_cP1;  //pots 1.5 inch panel
init_number set_Dmort_cP2;  //pots 2.0 inch panel

// Von Bert parameters in TL mm
init_number set_Linf;
init_number set_K;
init_number set_t0;
//Standard erros of von bert params
init_number set_Linf_se;
init_number set_K_se;
init_number set_t0_se;
//CV of length at age and its standard error
init_number set_len_cv;
init_number set_len_cv_se;
//TL(mm)-weight(whole weight in g) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;
//weight(whole weight)- fecundity (units=eggs/batch) relationship: log(y)=a+bW
init_number fecpar_a;
init_number fecpar_b;
init_number fecpar_batches;  //number of annual batches may be important if age-dependent, otherwise just a scalar
init_number fecpar_scale;    //used for scaling annual egg production (10^X eggs)

//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age
init_number spawn_time_frac; //time of year of peak spawning, as a fraction of the year

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

```
init_number set_M_constant_se;  //SE of age-independent M, used in prior, if M is estimated
init_number max_obs_age;        //max observed age, used to scale M


//Spawner-recruit parameters (Initial guesses or fixed values)
init_number set_steep;         //recruitment steepness
init_number set_steep_se;      //SE of recruitment steepness
init_int steep_prior_pdf;      //(1=none, 2=lognormal, 3=normal, 4=beta)
init_number set_log_R0;        //recruitment R0
init_number set_R_autocorr;    //recruitment autocorrelation
init_number set_rec_sigma;     //recruitment standard deviation in log space
init_number set_rec_sigma_se;  //SE of recruitment standard deviation in log space
init_int rec_sigma_prior_pdf;      //(1=none, 2=lognormal, 3=normal, 4=beta)



//Initial guesses or fixed values of estimated selectivity parameters
init_number set_selpar_L50_Mbft;
init_number set_selpar_slope_Mbft;

init_number set_selpar_L50_Mcvt;
init_number set_selpar_slope_Mcvt;

init_number set_selpar_L50_cL2;
init_number set_selpar_slope_cL2;
init_number set_selpar_L50_cL3;
init_number set_selpar_slope_cL3;

init_number set_selpar_L50_cP2;
init_number set_selpar_slope_cP2;
init_number set_selpar_L50_cP3;
init_number set_selpar_slope_cP3;

init_number set_selpar_L50_HB1;
init_number set_selpar_slope_HB1;
init_number set_selpar_L50_HB2;
init_number set_selpar_slope_HB2;
init_number set_selpar_L50_HB3;
init_number set_selpar_slope_HB3;
init_number set_selpar_L50_HB4;
init_number set_selpar_slope_HB4;

init_number set_selpar_L50_mrip1;
init_number set_selpar_slope_mrip1;
init_number set_selpar_L50_mrip2;
init_number set_selpar_slope_mrip2;
init_number set_selpar_L50_mrip3;
init_number set_selpar_slope_mrip3;
init_number set_selpar_L50_mrip4;
init_number set_selpar_slope_mrip4;


init_number set_selpar_Age0_HB_D_logit;
init_number set_selpar_Age1_HB_D_logit;
init_number set_selpar_Age2_HB_D_logit;



//--weights for likelihood components-----------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_D;

init_number set_w_lc_Mbft;
init_number set_w_lc_cL;
init_number set_w_lc_cP;
init_number set_w_lc_HB;
init_number set_w_lc_HB_D;
init_number set_w_lc_mrip;

init_number set_w_ac_Mbft;
init_number set_w_ac_Mcvt;
init_number set_w_ac_cL;
init_number set_w_ac_cP;
init_number set_w_ac_HB;
init_number set_w_ac_mrip;

init_number set_w_I_Mbft;
init_number set_w_I_Mcvt;
init_number set_w_I_cL;
init_number set_w_I_HB;
init_number set_w_I_HBD;

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

//value or initial guess for recreational HB and mrip historic landings multiplicative bias, last yr the bias applies
//this feature is not currently implemented in the likelihood fcn
init_number set_L_hb_bias;
init_number set_L_mrip_bias;
init_number set_L_comm_bias;
init_number endyr_L_HB_bias;
init_number endyr_L_mrip_bias;
init_number endyr_L_comm_bias;

//--index catchability--------------------------------------------------------------------------------------------------------
init_number set_logq_Mbft;    //catchability coefficient (log) for HBD
init_number set_logq_Mcvt;    //catchability coefficient (log) for HBD
init_number set_logq_cL;      //catchability coefficient (log) for commercial logbook index
init_number set_logq_HB;      //catchability coefficient (log) for the headboat index
```

```
init_number set_logq_HBD;     //catchability coefficient (log) for HBD

//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_cL_var;     //assumed variance of RW q
init_number set_q_RW_HB_var;     //assumed variance of RW q
init_number set_q_RW_HBD_var;    //assumed variance of RW q


////--F's---------------------------------
init_number set_log_avg_F_cL;
init_number set_log_avg_F_cP;
init_number set_log_avg_F_cT;
init_number set_log_avg_F_HB;
init_number set_log_avg_F_mrip;

init_number set_F_init_ratio;  //defines initialization F as a ratio of that from first several yrs of assessment

////--discard F's----------------------
init_number set_log_avg_F_comm_D;
init_number set_log_avg_F_HB_D;
init_number set_log_avg_F_mrip_D;


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


//threshold sample sizes for including length comps, age comps, respectively
init_number minSS_lenc;
init_number minSS_agec;

//maximum allowable annual sample sizes for length comps, age comps, respectively
init_number maxSS_lenc;
init_number maxSS_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);

//proportion of length comp mass below size limit considered when matching length comp
//note: these need length comp and age comp data to be estimable
init_number set_p_lenc_cL2;
init_number set_p_lenc_cL3;
init_number set_p_lenc_cP2;
init_number set_p_lenc_cP3;
init_number set_p_lenc_cT2;
init_number set_p_lenc_cT3;
init_number set_p_lenc_HB2;
init_number set_p_lenc_HB3;
init_number set_p_lenc_HB4;
init_number set_p_lenc_mrip2;
init_number set_p_lenc_mrip3;
init_number set_p_lenc_mrip4;

init_number set_p_lenc_comm_D2;
init_number set_p_lenc_comm_D3;
init_number set_p_lenc_HB_D2;
init_number set_p_lenc_HB_D3;
init_number set_p_lenc_HB_D4;
init_number set_p_lenc_mrip_D1;
init_number set_p_lenc_mrip_D2;
init_number set_p_lenc_mrip_D3;
init_number set_p_lenc_mrip_D4;

// #######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
number onehalf;                 //0.5

init_number end_of_data_file;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   if(end_of_data_file!=999)
   {
     for(iyear=1; iyear<=1000; iyear++)
     {
       cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
       cout << "" <<endl;
     }
   }
   else
   {
    cout << "Data File read correctly" << endl;
```

```
   }
 END_CALCS


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

  //init_bounded_number Linf(300,800,3);
  //init_bounded_number K(0.05,0.5,3);
  //init_bounded_number t0(-1.5,-0.01,3);
  number Linf;
  number K;
  number t0;
  init_bounded_number len_cv_val(0.07,0.3,4);
  //  init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,-4)
  //number len_cv_val;
  vector len_sd(1,nages);
  vector len_cv(1,nages);
  vector meanlen_TL(1,nages);   //mean Total length (mm) at age
  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 fecundity(1,nages);    //fecundity at age, perhaps scaled

  matrix len_cL_mm(styr,endyr,1,nages);       //mean length at age of cL landings in mm (may differ from popn mean)
  matrix wgt_cL_klb(styr,endyr,1,nages);      //whole wgt of cL landings in 1000 lb
  matrix len_cP_mm(styr,endyr,1,nages);       //mean length at age of cP landings in mm (may differ from popn mean)
  matrix wgt_cP_klb(styr,endyr,1,nages);      //whole wgt of cP landings in 1000 lb
  matrix len_cT_mm(styr,endyr,1,nages);       //mean length at age of cO landings in mm (may differ from popn mean)
  matrix wgt_cT_klb(styr,endyr,1,nages);      //whole wgt of cO landings in 1000 lb
  matrix len_HB_mm(styr,endyr,1,nages);       //mean length at age of HB landings in mm (may differ from popn mean)
  matrix wgt_HB_klb(styr,endyr,1,nages);      //whole wgt of HB landings in 1000 lb
  matrix len_mrip_mm(styr,endyr,1,nages);     //mean length at age of mrip landings in mm (may differ from popn mean)
  matrix wgt_mrip_klb(styr,endyr,1,nages);    //whole wgt of mrip landings in 1000 lb

  matrix len_comm_D_mm(styr,endyr,1,nages);    //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_comm_D_klb(styr,endyr,1,nages);   //whole wgt of cL discards in 1000 lb
  matrix len_HB_D_mm(styr,endyr,1,nages);      //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_HB_D_klb(styr,endyr,1,nages);     //whole wgt of cL discards in 1000 lb
  matrix len_mrip_D_mm(styr,endyr,1,nages);    //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_mrip_D_klb(styr,endyr,1,nages);   //whole wgt of cL discards in 1000 lb

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

  //matrices below are used to match length comps
  matrix lenprob_Mbft(1,nages,1,nlenbins);    //distn of size at age of Mbft
  matrix lenprob_cL1(1,nages,1,nlenbins);     //distn of size at age in cL block 1
  matrix lenprob_cL2(1,nages,1,nlenbins);     //distn of size at age in cL block 2
  matrix lenprob_cL3(1,nages,1,nlenbins);     //distn of size at age in cL block 3
  matrix lenprob_cP1(1,nages,1,nlenbins);     //distn of size at age in cP block 1
  matrix lenprob_cP2(1,nages,1,nlenbins);     //distn of size at age in cP block 2
  matrix lenprob_cP3(1,nages,1,nlenbins);     //distn of size at age in cP block 3
  matrix lenprob_cT1(1,nages,1,nlenbins);     //distn of size at age in cT block 1
  matrix lenprob_cT2(1,nages,1,nlenbins);     //distn of size at age in cT block 2
  matrix lenprob_HB1(1,nages,1,nlenbins);     //distn of size at age in HB block 1
  matrix lenprob_HB2(1,nages,1,nlenbins);     //distn of size at age in HB block 2
  matrix lenprob_HB3(1,nages,1,nlenbins);     //distn of size at age in HB block 3
  matrix lenprob_HB4(1,nages,1,nlenbins);     //distn of size at age in HB block 4
  matrix lenprob_mrip1(1,nages,1,nlenbins);   //distn of size at age in mrip block 2
  matrix lenprob_mrip2(1,nages,1,nlenbins);   //distn of size at age in mrip block 2
  matrix lenprob_mrip3(1,nages,1,nlenbins);   //distn of size at age in mrip block 3
  matrix lenprob_mrip4(1,nages,1,nlenbins);   //distn of size at age in mrip block 4

  matrix lenprob_comm_D2(1,nages,1,nlenbins);    //distn of size at age in comm discards comm block 2
  matrix lenprob_comm_D3(1,nages,1,nlenbins);    //distn of size at age in comm discards comm block 3
  matrix lenprob_HB_D2(1,nages,1,nlenbins);      //distn of size at age in HB discards rec block 2
  matrix lenprob_HB_D3(1,nages,1,nlenbins);      //distn of size at age in HB discards rec block 3
  matrix lenprob_HB_D4(1,nages,1,nlenbins);      //distn of size at age in HB discards rec block 4
  matrix lenprob_mrip_D1(1,nages,1,nlenbins);    //distn of size at age in mrip discards rec block 1
  matrix lenprob_mrip_D2(1,nages,1,nlenbins);    //distn of size at age in mrip discards rec block 2
  matrix lenprob_mrip_D3(1,nages,1,nlenbins);    //distn of size at age in mrip discards rec block 3
  matrix lenprob_mrip_D4(1,nages,1,nlenbins);    //distn of size at age in mrip discards rec block 4

  //matrices below used to compute mean weights
  matrix lenprob_cL1_all(1,nages,1,nlenbins_all);     //distn of size at age in cL block 1
  matrix lenprob_cL2_all(1,nages,1,nlenbins_all);     //distn of size at age in cL block 2
  matrix lenprob_cL3_all(1,nages,1,nlenbins_all);     //distn of size at age in cL block 3
  matrix lenprob_cP1_all(1,nages,1,nlenbins_all);     //distn of size at age in cP block 1
  matrix lenprob_cP2_all(1,nages,1,nlenbins_all);     //distn of size at age in cP block 2
  matrix lenprob_cP3_all(1,nages,1,nlenbins_all);     //distn of size at age in cP block 3
  matrix lenprob_cT1_all(1,nages,1,nlenbins_all);     //distn of size at age in cT block 1
  matrix lenprob_cT2_all(1,nages,1,nlenbins_all);     //distn of size at age in cT block 2
  matrix lenprob_HB1_all(1,nages,1,nlenbins_all);     //distn of size at age in HB block 1
  matrix lenprob_HB2_all(1,nages,1,nlenbins_all);     //distn of size at age in HB block 2
  matrix lenprob_HB3_all(1,nages,1,nlenbins_all);     //distn of size at age in HB block 3
  matrix lenprob_HB4_all(1,nages,1,nlenbins_all);     //distn of size at age in HB block 4
  matrix lenprob_mrip1_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip block 1
  matrix lenprob_mrip2_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip block 2
  matrix lenprob_mrip3_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip block 3
  matrix lenprob_mrip4_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip block 4

  matrix lenprob_comm_D2_all(1,nages,1,nlenbins_all);   //distn of size at age in cL discards comm block 2
  matrix lenprob_comm_D3_all(1,nages,1,nlenbins_all);   //distn of size at age in cL discards comm block 3
  matrix lenprob_HB_D2_all(1,nages,1,nlenbins_all);     //distn of size at age in HB discards rec block 2
  matrix lenprob_HB_D3_all(1,nages,1,nlenbins_all);     //distn of size at age in HB discards rec block 3
  matrix lenprob_HB_D4_all(1,nages,1,nlenbins_all);     //distn of size at age in HB discards rec block 4
  matrix lenprob_mrip_D1_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip discards rec block 1
```

```
  matrix lenprob_mrip_D2_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip discards rec block 2
  matrix lenprob_mrip_D3_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip discards rec block 3
  matrix lenprob_mrip_D4_all(1,nages,1,nlenbins_all);   //distn of size at age in mrip discards rec block 4

//----Predicted length and age compositions
  matrix pred_Mbft_lenc(1,nyr_Mbft_lenc,1,nlenbins);
  matrix pred_cL_lenc(1,nyr_cL_lenc,1,nlenbins);
  matrix pred_cP_lenc(1,nyr_cP_lenc,1,nlenbins);
  matrix pred_HB_lenc(1,nyr_HB_lenc,1,nlenbins);
  matrix pred_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);
  matrix pred_mrip_lenc(1,nyr_mrip_lenc,1,nlenbins);

  matrix L_cP_num_pool(1,nyr_cP_lenc,1,nages);          //landings (numbers) at age pooled for length comps
  matrix L_cP_num_pool_yr(1,nyr_cP_lenc_pool,1,nages);  //scaled and weighted landings (numbers) for pooling length comps
  //matrix L_mrip_num_pool(1,nyr_mrip_lenc,1,nages);          //landings (numbers) at age pooled for length comps
  //matrix L_mrip_num_pool_yr(1,nyr_mrip_lenc_pool,1,nages);  //scaled and weighted landings (numbers) for pooling length comps

//  //##p_lenc_fishery pars require age comp and length comp data for estimation
  number p_lenc_cL2;
  number p_lenc_cL3;
  number p_lenc_cP2;
  number p_lenc_cP3;
  number p_lenc_cT2;
  number p_lenc_cT3;
  number p_lenc_HB2;
  number p_lenc_HB3;
  number p_lenc_HB4;
  number p_lenc_mrip2;
  number p_lenc_mrip3;
  number p_lenc_mrip4;

  number p_lenc_comm_D2;
  number p_lenc_comm_D3;
  number p_lenc_HB_D2;
  number p_lenc_HB_D3;
  number p_lenc_HB_D4;
  number p_lenc_mrip_D1;
  number p_lenc_mrip_D2;
  number p_lenc_mrip_D3;
  number p_lenc_mrip_D4;

  matrix pred_Mbft_agec(1,nyr_Mbft_agec,1,nages);
  matrix ErrorFree_Mbft_agec(1,nyr_Mbft_agec,1,nages);
  matrix pred_Mcvt_agec(1,nyr_Mcvt_agec,1,nages);
  matrix ErrorFree_Mcvt_agec(1,nyr_Mcvt_agec,1,nages);
  matrix pred_cL_agec(1,nyr_cL_agec,1,nages);
  matrix ErrorFree_cL_agec(1,nyr_cL_agec,1,nages);
  matrix pred_cP_agec(1,nyr_cP_agec,1,nages);
  matrix ErrorFree_cP_agec(1,nyr_cP_agec,1,nages);
  matrix pred_HB_agec(1,nyr_HB_agec,1,nages);
  matrix ErrorFree_HB_agec(1,nyr_HB_agec,1,nages);
  matrix pred_mrip_agec(1,nyr_mrip_agec,1,nages);
  matrix ErrorFree_mrip_agec(1,nyr_mrip_agec,1,nages);

//effective sample size applied in multinomial distributions
  vector nsamp_Mbft_lenc_allyr(styr,endyr);
  vector nsamp_cL_lenc_allyr(styr,endyr);
  vector nsamp_cP_lenc_allyr(styr,endyr);
  vector nsamp_HB_lenc_allyr(styr,endyr);
  vector nsamp_HB_D_lenc_allyr(styr,endyr);
  vector nsamp_mrip_lenc_allyr(styr,endyr);
  vector nsamp_Mbft_agec_allyr(styr,endyr);
  vector nsamp_Mcvt_agec_allyr(styr,endyr);
  vector nsamp_cL_agec_allyr(styr,endyr);
  vector nsamp_cP_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);
  vector nsamp_mrip_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_Mbft_lenc_allyr(styr,endyr);
  vector nfish_cL_lenc_allyr(styr,endyr);
  vector nfish_cP_lenc_allyr(styr,endyr);
  vector nfish_HB_lenc_allyr(styr,endyr);
  vector nfish_HB_D_lenc_allyr(styr,endyr);
  vector nfish_mrip_lenc_allyr(styr,endyr);
  vector nfish_Mbft_agec_allyr(styr,endyr);
  vector nfish_Mcvt_agec_allyr(styr,endyr);
  vector nfish_cL_agec_allyr(styr,endyr);
  vector nfish_cP_agec_allyr(styr,endyr);
  vector nfish_HB_agec_allyr(styr,endyr);
  vector nfish_mrip_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_Mbft_lenc_allyr_out(styr,endyr);
  vector neff_cL_lenc_allyr_out(styr,endyr);
  vector neff_cP_lenc_allyr_out(styr,endyr);
  vector neff_HB_lenc_allyr_out(styr,endyr);
  vector neff_HB_D_lenc_allyr_out(styr,endyr);
  vector neff_mrip_lenc_allyr_out(styr,endyr);
  vector neff_Mbft_agec_allyr_out(styr,endyr);
  vector neff_Mcvt_agec_allyr_out(styr,endyr);
  vector neff_cL_agec_allyr_out(styr,endyr);
  vector neff_cP_agec_allyr_out(styr,endyr);
  vector neff_HB_agec_allyr_out(styr,endyr);
  vector neff_mrip_agec_allyr_out(styr,endyr);


//-----Population---------------------------------------------------------------------------------
  matrix N(styr,endyr,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
  init_bounded_vector log_Nage_dev(2,nages,-5,3,1); //log deviations on initial abundance at age
  //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,nages);            //Population biomass by year and age at start of yr
  vector totB(styr,endyr);                 //Total biomass by year
  vector totN(styr,endyr);                 //Total abundance by year
  vector SSB(styr,endyr);                  //Total spawning biomass by year (scaled popn fecundity)
  vector MatFemB(styr,endyr);              //Total spawning biomass by year (total mature female biomass)
  vector rec(styr,endyr);                  //Recruits by year
  vector prop_f(1,nages);                  //Proportion female by age
  vector maturity_f(1,nages);              //Proportion of female mature at age
  vector reprod(1,nages);                  //vector used to compute spawning biomass (scaled popn fecundity)
  vector reprod2(1,nages);                 //vector used to compute mature female biomass

////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(13,20,1);       //log(virgin Recruitment)
  //number log_R0;
  number R0;                               //virgin recruitment
  init_bounded_number steep(0.21,0.991,3);   //steepness
  //number steep;  //uncomment to fix steepness, comment line directly above
  init_bounded_number rec_sigma(0.1,1.5,4);  //sd recruitment residuals
  number rec_sigma_sq;                     //square of rec_sigma
  number rec_sigma_sqd2;                   //square of rec_sigma divided by two
  number rec_logL_add;                     //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr,-3,3,2);  //log recruitment deviations
  //vector log_rec_dev(styr_rec_dev,endyr);
  vector log_rec_dev_output(styr,endyr);           //used in output. equals zero except for yrs in log_rec_dev

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

  number BiasCor;                                //Bias correction in equilibrium recruits
  //init_bounded_number R_autocorr(-1.0,1.0,2);  //autocorrelation in SR
  number R_autocorr;
  number S0;                                     //equal to spr_F0*R0 = virgin SSB
  number B0;                                     //equal to bpr_F0*R0 = virgin B
  number R1;                                     //Recruits in styr
  number R_virgin;                               //unfished recruitment with bias correction
  vector SdS0(styr,endyr);                       //SSB / virgin SSB


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

//MARMAP Mbft selectivity ------------------------------------------------------------------------
  init_bounded_number selpar_L50_Mbft(0.5,8.0,1);
  init_bounded_number selpar_slope_Mbft(0.1,10.0,1);

  vector sel_Mbft_vec(1,nages);
  matrix sel_Mbft(styr,endyr,1,nages);

//MARMAP Mcvt selectivity ------------------------------------------------------------------------
  init_bounded_number selpar_L50_Mcvt(0.5,8.0,1);
  init_bounded_number selpar_slope_Mcvt(0.1,10.0,1);
  vector sel_Mcvt_vec(1,nages);
  matrix sel_Mcvt(styr,endyr,1,nages);

//Commercial handline selectivity----------------------------------------------------
  init_bounded_number selpar_L50_cL2(0.5,8.0,1);
  init_bounded_number selpar_slope_cL2(0.1,10.0,1);

  init_bounded_number selpar_L50_cL3(0.5,8.0,1);
  init_bounded_number selpar_slope_cL3(0.1,10.0,1);

  //vector sel_cL_1(1,nages); //sel in period 1 assumed equal to period 2
  vector sel_cL_2(1,nages); //sel in period 2
  vector sel_cL_3(1,nages); //sel in period 3
  matrix sel_cL(styr,endyr,1,nages);
  //commercial discards (handline + pots)
  vector sel_comm_D_2(1,nages);         //sel in period 2
  vector sel_comm_D_3(1,nages);         //sel in period 3
  vector sel_comm_D_quota3(1,nages);    //sel in period 3 when quotas were in place (2009,2010)
  matrix sel_comm_D(styr,endyr,1,nages);
  //values used for weighting selex and avg weights of discards during yrs with quotas
  number Dopen_cL; number Dclosed_cL; number Lopen_cL;
  number Dopen_cP; number Dclosed_cP; number Lopen_cP;
  number D_sum_cLcP;
  number Dprop_comm_sel_D; number Dprop_comm_sel_cL; number Dprop_comm_sel_cP;

//Commercial pots selectivity --------------------------------------------------
  init_bounded_number selpar_L50_cP2(0.5,8.0,1);
  init_bounded_number selpar_slope_cP2(0.1,10.0,1);

  init_bounded_number selpar_L50_cP3(0.5,8.0,1);
  init_bounded_number selpar_slope_cP3(0.1,10.0,1);

//  vector sel_cP_1(1,nages); //sel vector in period 1 assumed equal to period 2
  vector sel_cP_2(1,nages);   //sel vector in period 2
  vector sel_cP_3(1,nages);   //sel vector in period 3
  matrix sel_cP(styr,endyr,1,nages);

//Commercial trawl selectivity --------------------------------------------------
  matrix sel_cT(styr,endyr,1,nages);  //mirrors comm pot sel

//Headboat selectivity ----------------------------------------------
  init_bounded_number selpar_L50_HB1(0.5,8.0,1);
  init_bounded_number selpar_slope_HB1(0.1,10.0,1);

  init_bounded_number selpar_L50_HB2(0.5,8.0,1);
  init_bounded_number selpar_slope_HB2(0.1,10.0,1);

  init_bounded_number selpar_L50_HB3(0.5,8.0,1);
  init_bounded_number selpar_slope_HB3(0.1,10.0,1);

  init_bounded_number selpar_L50_HB4(0.5,8.0,1);
  init_bounded_number selpar_slope_HB4(0.1,10.0,1);
```

```
  vector sel_HB_1(1,nages); //sel in period 1
  vector sel_HB_2(1,nages); //sel in period 2
  vector sel_HB_3(1,nages); //sel in period 3
  vector sel_HB_4(1,nages); //sel in period 4
  matrix sel_HB(styr,endyr,1,nages);

  //---headboat discards--------------------------------
  vector sel_HB_D_1(1,nages); //sel in period 1, assumed equal to period 2
  vector sel_HB_D_2(1,nages); //sel in period 2
  vector sel_HB_D_3(1,nages); //sel in period 3
  vector sel_HB_D_4(1,nages); //sel in period 4
  matrix sel_HB_D(styr,endyr,1,nages);

  vector vecprob_HB_D2(4,nages);     //prob of less than size limit
  vector vecprob_HB_D3(4,nages);     //prob of less than size limit
  vector vecprob_HB_D4(4,nages);     //prob of less than size limit

  init_bounded_number selpar_Age0_HB_D_logit(-15.0,10.0,1); //estimated in logit space: period2, period 3
  number selpar_Age0_HB_D;
  init_bounded_number selpar_Age1_HB_D_logit(-15.0,10.0,1); //estimated in logit space: period2, period 3
  number selpar_Age1_HB_D;
  init_bounded_number selpar_Age2_HB_D_logit(-15.0,10.0,1); //estimated in logit space: period2, period 3
  number selpar_Age2_HB_D;

//mrip selectivity  ------------------------------------------------
  //init_bounded_number selpar_L50_mrip1(0.5,8.0,1);
  //init_bounded_number selpar_slope_mrip1(0.1,10.0,1);
  number selpar_L50_mrip1;
  number selpar_slope_mrip1;

  //init_bounded_number selpar_L50_mrip2(0.5,8.0,1);
  //init_bounded_number selpar_slope_mrip2(0.1,10.0,1);
  number selpar_L50_mrip2;
  number selpar_slope_mrip2;

  //init_bounded_number selpar_L50_mrip3(0.5,8.0,1);
  //init_bounded_number selpar_slope_mrip3(0.1,10.0,1);
  number selpar_L50_mrip3;
  number selpar_slope_mrip3;

  //init_bounded_number selpar_L50_mrip4(0.5,8.0,1);
  //init_bounded_number selpar_slope_mrip4(0.1,10.0,1);
  number selpar_L50_mrip4;
  number selpar_slope_mrip4;

  vector sel_mrip_1(1,nages); //sel in period 1
  vector sel_mrip_2(1,nages); //sel in period 2
  vector sel_mrip_3(1,nages); //sel in period 3
  vector sel_mrip_4(1,nages); //sel in period 4
  matrix sel_mrip(styr,endyr,1,nages);
  matrix sel_mrip_D(styr,endyr,1,nages);


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

//-------------------------------------------------------------------------------------------------------------------------------------------------------
//-------CPUE Predictions--------------------------------
  vector pred_Mbft_cpue(styr_Mbft_cpue,endyr_Mbft_cpue);       //predicted Mbft U (fish/trap-hour)
  matrix N_Mbft(styr_Mbft_cpue,endyr_Mbft_cpue,1,nages);       //used to compute Mbft index
  vector pred_Mcvt_cpue(styr_Mcvt_cpue,endyr_Mcvt_cpue);       //predicted Mcvt U (fish/trap-hour)
  matrix N_Mcvt(styr_Mcvt_cpue,endyr_Mcvt_cpue,1,nages);       //used to compute Mcvt index
  vector pred_cL_cpue(styr_cL_cpue,endyr_cL_cpue);             //predicted cL U (pounds/hook-hour)
  matrix N_cL(styr_cL_cpue,endyr_cL_cpue,1,nages);             //used to compute cL index
  vector pred_HB_cpue(styr_HB_cpue,endyr_HB_cpue);             //predicted HB U (pounds/hour)
  matrix N_HB(styr_HB_cpue,endyr_HB_cpue,1,nages);             //used to compute HB index
  vector pred_HBD_cpue(styr_HBD_cpue,endyr_HBD_cpue);          //predicted HBD U (fish/angler-hour)
  matrix N_HBD(styr_HBD_cpue,endyr_HBD_cpue,1,nages);          //used to compute HBD index

//---Catchability (CPUE q's)-----------------------------------------------------
  init_bounded_number log_q_Mbft(-20,-10,1);
  init_bounded_number log_q_Mcvt(-20,-10,1);
  init_bounded_number log_q_cL(-20,-5,1);
  init_bounded_number log_q_HB(-20,-5,1);
  init_bounded_number log_q_HBD(-20,-10,1);
  init_bounded_number q_rate(0.001,0.1,set_q_rate_phase);
  //number q_rate;
  vector q_rate_fcn_cL(styr_cL_cpue,endyr_cL_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_HBD(styr_HBD_cpue,endyr_HBD_cpue);      //increase due to technology creep (saturates in 2003)

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

  init_bounded_vector q_RW_log_dev_cL(styr_cL_cpue,endyr_cL_cpue-1,-3.0,3.0,set_q_RW_phase);
  init_bounded_vector q_RW_log_dev_HB(styr_HB_cpue,endyr_HB_cpue-1,-3.0,3.0,set_q_RW_phase);
  init_bounded_vector q_RW_log_dev_HBD(styr_HBD_cpue,endyr_HBD_cpue-1,-3.0,3.0,set_q_RW_phase);

  vector q_cL(styr_cL_cpue,endyr_cL_cpue);
  vector q_HB(styr_HB_cpue,endyr_HB_cpue);
  vector q_HBD(styr_HBD_cpue,endyr_HBD_cpue);

//-------------------------------------------------------------------------------------------------------------------------------------------------------
//---Landings Bias for recreational landings-------------------------------------------------------------------
  //init_bounded_number L_mrip_bias(0.1,10.0,3);
  number L_hb_bias;
  number L_mrip_bias;
  number L_comm_bias;
```

```
//---Landings in numbers (total or 1000 fish) and in wgt (klb)-------------------------------------------------
  matrix L_cL_num(styr,endyr,1,nages);  //landings (numbers) at age
  matrix L_cL_klb(styr,endyr,1,nages);  //landings (1000 lb whole weight) at age
  vector pred_cL_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_cL_L_klb(styr,endyr);     //yearly landings in 1000 lb summed over ages

  matrix L_cP_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cP_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cP_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cP_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages

  matrix L_cT_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cT_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cT_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cT_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages

  matrix L_HB_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_HB_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_HB_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_HB_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages
  vector obs_HB_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicative bias

  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
  vector obs_mrip_L_wbias(styr,endyr);    //yearly landings observed, perhaps adjusted for multiplicative bias

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

//---Dead discards in numbers (total or 1000 fish) and in wgt (klb) -------------------------------------------
  matrix D_comm_num(styr,endyr,1,nages);   //discards (numbers) at age
  matrix D_comm_klb(styr,endyr,1,nages);   //discards (1000 lb) at age
  vector pred_comm_D_knum(styr,endyr);     //yearly discards summed over ages
  vector pred_comm_D_klb(styr,endyr);      //yearly discards in klb summed over ages
  vector obs_comm_D(styr_cL_D,endyr_cL_D); //observed releases multiplied by discard mortality
  vector obs_cL_D(styr_cL_D,endyr_cL_D);   //observed releases multiplied by discard mortality
  vector obs_cP_D(styr_cL_D,endyr_cL_D);   //observed releases multiplied by discard mortality
  vector comm_D_cv(styr_cL_D,endyr_cL_D);  //CVs for fitting combined discards

  matrix D_HB_num(styr,endyr,1,nages);     //discards (numbers) at age
  matrix D_HB_klb(styr,endyr,1,nages);     //discards (1000 lb) at age
  vector pred_HB_D_knum(styr,endyr);       //yearly discards summed over ages
  vector pred_HB_D_klb(styr,endyr);        //yearly discards in klb summed over ages
  vector obs_HB_D(styr_HB_D,endyr_HB_D);   //observed releases multiplied by discard mortality

  matrix D_mrip_num(styr,endyr,1,nages);   //discards (numbers) at age
  matrix D_mrip_klb(styr,endyr,1,nages);   //discards (1000 lb) at age
  vector pred_mrip_D_knum(styr,endyr);     //yearly discards summed over ages
  vector pred_mrip_D_klb(styr,endyr);      //yearly discards in klb summed over ages
  vector obs_mrip_D(styr_mrip_D,endyr_mrip_D); //observed releases multiplied by discard mortality

  matrix D_total_num(styr,endyr,1,nages);  //total discards in number at age
  matrix D_total_klb(styr,endyr,1,nages);  //discards in klb at age
  vector D_total_knum_yr(styr,endyr);      //total discards in 1000 fish by yr summed over ages
  vector D_total_klb_yr(styr,endyr);       //total discards (klb) by yr summed over ages

////---MSY calcs-------------------------------------------------------------------------------
  number F_cL_prop;       //proportion of F_sum attributable to hal, last X=selpar_n_yrs_wgted yrs, used for avg body weights
  number F_cP_prop;       //proportion of F_sum attributable to pots, last X yrs
  number F_HB_prop;       //proportion of F_sum attributable to headboat, last X yrs
  number F_mrip_prop;     //proportion of F_sum attributable to mrip, last X yrs
  number F_comm_D_prop;   //proportion of F_sum attributable to comm discards, last X yrs
  number F_HB_D_prop;     //proportion of F_sum attributable to headboat discards, last X yrs
  number F_mrip_D_prop;   //proportion of F_sum attributable to mrip discards, last X yrs
  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);
  vector F_end_D(1,nages);
  number F_end_apex;

  number SSB_msy_out;           //SSB (popn fecudity) 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 D_msy_knum_out;        //equilibrium dead discards (1000 fish) at F=Fmsy
  number D_msy_klb_out;         //equilibrium dead discards (1000 lb) 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: time of peak spawning
  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 D_age_msy(1,nages);         //discard mortality (dead discards) at age for MSY calculations
  vector F_L_age_msy(1,nages);       //fishing mortality landings (not discards) at age for MSY calculations
  vector F_D_age_msy(1,nages);       //fishing mortality of 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 D_eq_klb(1,n_iter_msy);     //equilibrium discards (klb) corresponding to F values in F_msy
  vector D_eq_knum(1,n_iter_msy);    //equilibrium discards (1000s) 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
  vector wgt_wgted_D_klb(1,nages);  //fishery-weighted average weight at age of discards
  number wgt_wgted_L_denom;         //used in intermediate calculations
  number wgt_wgted_D_denom;         //used in intermediate calculations

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

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

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

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

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

  init_bounded_number log_avg_F_cL(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cL(styr_cL_L,endyr_cL_L,-10.0,5.0,2);
  matrix F_cL(styr,endyr,1,nages);
  vector F_cL_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cL;
  number log_F_dev_end_cL;

  init_bounded_number log_avg_F_cP(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cP(styr_cP_L,endyr_cP_L,-10.0,5.0,2);
  matrix F_cP(styr,endyr,1,nages);
  vector F_cP_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cP;
  number log_F_dev_end_cP;

  init_bounded_number log_avg_F_cT(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cT(styr_cT_L,endyr_cT_L,-10.0,5.0,2);
  matrix F_cT(styr,endyr,1,nages);
  vector F_cT_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cT;
  number log_F_dev_end_cT;

  init_bounded_number log_avg_F_HB(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_HB(styr_HB_L,endyr_HB_L,-10.0,5.0,2);
  matrix F_HB(styr,endyr,1,nages);
  vector F_HB_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_init_HB;
  number log_F_dev_end_HB;

  init_bounded_number log_avg_F_mrip(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_mrip(styr_mrip_L,endyr_mrip_L,-10.0,5.0,2);
  matrix F_mrip(styr,endyr,1,nages);
  vector F_mrip_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_mrip;
  number log_F_dev_end_mrip;

  init_bounded_number F_init_ratio(0.1,1.5,1); //scales initial F, which is geometric mean first three yrs
  //number F_init_ratio;

//--Discard mortality stuff--------------------------------------------------------------------------
  init_bounded_number log_avg_F_comm_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_comm_D(styr_cL_D,endyr_cL_D,-10.0,5.0,2);
  matrix F_comm_D(styr,endyr,1,nages);
  vector F_comm_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_comm_D2;   //avg log deviations in reg period 2 (for estimation 1984-1992, prior to data)
  number log_F_dev_end_comm_D;

  init_bounded_number log_avg_F_HB_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_HB_D(styr_HB_D,endyr_HB_D,-10.0,5.0,2);
  matrix F_HB_D(styr,endyr,1,nages);
  vector F_HB_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_HB_D;

  init_bounded_number log_avg_F_mrip_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_mrip_D(styr_mrip_D,endyr_mrip_D,-10.0,5.0,2);
  matrix F_mrip_D(styr,endyr,1,nages);
  vector F_mrip_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_mrip_D;
  number log_F_dev_end_mrip_D;

  number Dmort_HL;
  number Dmort_cP1;
  number Dmort_cP2;

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

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

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


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

  number w_lc_Mbft;
  number w_lc_cL;
  number w_lc_cP;
  number w_lc_HB;
  number w_lc_HB_D;
  number w_lc_mrip;

  number w_ac_Mbft;
  number w_ac_Mcvt;
  number w_ac_cL;
  number w_ac_cP;
  number w_ac_HB;
  number w_ac_mrip;

  number w_I_Mbft;
  number w_I_Mcvt;
  number w_I_cL;
  number w_I_HB;
  number w_I_HBD;

  number w_rec;
  number w_rec_early;
  number w_rec_end;
  number w_fullF;
  number w_Ftune;
//  number w_cvlen_dev;
//  number w_cvlen_diff;

  number f_Mbft_cpue;
  number f_Mcvt_cpue;
  number f_cL_cpue;
  number f_HB_cpue;
  number f_HBD_cpue;

  number f_cL_L;
  number f_cP_L;
  number f_cT_L;
  number f_HB_L;
  number f_mrip_L;

  number f_comm_D;
  number f_HB_D;
  number f_mrip_D;

  number f_Mbft_lenc;
  number f_cL_lenc;
  number f_cP_lenc;
  number f_HB_lenc;
  number f_HB_D_lenc;
  number f_mrip_lenc;

  number f_Mbft_agec;
  number f_Mcvt_agec;
  number f_cL_agec;
  number f_cP_agec;
  number f_HB_agec;
  number f_mrip_agec;

  number f_cL_RW_cpue; //random walk component of indices
  number f_HB_RW_cpue;
  number f_HBD_RW_cpue;

//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_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
  //number f_cvlen_dev_constraint; //deviation penalty on cv's of length at age
  //number f_cvlen_diff_constraint;//first diff penalty on cv's of length at age

  objective_function_value fval;
  number fval_data;

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

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


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
```

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

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

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

//// Set values of fixed parameters or set initial guess of estimated parameters
 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;
 onehalf=0.5;

  Dmort_HL=set_Dmort_HL;
  Dmort_cP1=set_Dmort_cP1;
  Dmort_cP2=set_Dmort_cP2;

  //values used for weighting selex and avg weight if comm discards in yrs with quotas
  //geometric mean of last two yrs
  //avg weights of landings were near 1 lb, so those values are left in weight
  Dopen_cL=Dmort_HL*pow((obs_cL_released(styr_comm_closed)*obs_cL_released(endyr)),onehalf);
  Dclosed_cL=Dmort_HL*pow((obs_cL_closed_released(styr_comm_closed)*obs_cL_closed_released(endyr)),onehalf);
  Lopen_cL=Dmort_HL*pow((obs_cL_L(styr_comm_closed)*obs_cL_L(endyr)),onehalf);
  Dopen_cP=Dmort_cP2*pow((obs_cP_released(styr_comm_closed)*obs_cP_released(endyr)),onehalf);
  Dclosed_cP=Dmort_cP2*pow((obs_cP_closed_released(styr_comm_closed)*obs_cP_closed_released(endyr)),onehalf);
  Lopen_cP=Dmort_cP2*pow((obs_cP_L(styr_comm_closed)*obs_cP_L(endyr)),onehalf);
  D_sum_cLcP=Dopen_cL+Dclosed_cL+Dopen_cP+Dclosed_cP;

  Dprop_comm_sel_D=(Dopen_cL + Dopen_cP + Dclosed_cL*(Dopen_cL/(Dopen_cL+Lopen_cL)) +
                   Dclosed_cP*(Dopen_cP/(Dopen_cP+Lopen_cP)))/D_sum_cLcP;
  Dprop_comm_sel_cL=Dclosed_cL*(Lopen_cL/(Dopen_cL+Lopen_cL))/D_sum_cLcP;
  Dprop_comm_sel_cP=Dclosed_cP*(Lopen_cP/(Dopen_cP+Lopen_cP))/D_sum_cLcP;;

  //discards values for fitting, include discard mortality
  obs_cL_D=Dmort_HL*obs_cL_released;
  obs_cL_D(styr_cL_closed_D,endyr_cL_closed_D)+=Dmort_HL*obs_cL_closed_released;
  obs_cP_D(styr_cP_D,2006)=Dmort_cP1*obs_cP_released(styr_cP_D,2006);
  obs_cP_D(2007,endyr_cP_D)=Dmort_cP2*obs_cP_released(2007,endyr_cP_D);
  obs_cP_D(styr_cP_closed_D,endyr_cP_closed_D)+=Dmort_cP2*obs_cP_closed_released;
  obs_comm_D=obs_cL_D+obs_cP_D;
  obs_HB_D=Dmort_HL*obs_HB_released;
  obs_mrip_D=Dmort_HL*obs_mrip_released;
  comm_D_cv=cL_D_cv;

  Linf=set_Linf;
  K=set_K;
  t0=set_t0;

//  age_limit_8in=t0-log(1.0-limit_8in/Linf)/K; //age at size limit: 8" limit;
//  age_limit_10in=t0-log(1.0-limit_10in/Linf)/K; //age at size limit: 10" limit;
//  age_limit_12in=t0-log(1.0-limit_12in/Linf)/K; //age at size limit: 12" limit;

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


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

  log_q_Mbft=set_logq_Mbft;
  log_q_Mcvt=set_logq_Mcvt;
  log_q_cL=set_logq_cL;
  log_q_HB=set_logq_HB;
  log_q_HBD=set_logq_HBD;
  q_rate=set_q_rate;
  q_rate_fcn_cL=1.0;
  q_rate_fcn_HB=1.0;
  q_rate_fcn_HBD=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;
  q_RW_log_dev_cL.initialize();
  q_RW_log_dev_HB.initialize();
  q_RW_log_dev_HBD.initialize();

  if (set_q_rate_phase<0 & q_rate!=0.0)
  {
      for (iyear=styr_cL_cpue; iyear<=endyr_cL_cpue; iyear++)
      {   if (iyear>styr_cL_cpue & iyear <=2003)
          {//q_rate_fcn_cL(iyear)=(1.0+q_rate)*q_rate_fcn_cL(iyear-1); //compound
             q_rate_fcn_cL(iyear)=(1.0+(iyear-styr_cL_cpue)*q_rate)*q_rate_fcn_cL(styr_cL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cL(iyear)=q_rate_fcn_cL(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_HBD_cpue; iyear<=endyr_HBD_cpue; iyear++)
```

```
    {   if (iyear>styr_HBD_cpue & iyear <=2003)
        {//q_rate_fcn_HBD(iyear)=(1.0+q_rate)*q_rate_fcn_HBD(iyear-1); //compound
           q_rate_fcn_HBD(iyear)=(1.0+(iyear-styr_HBD_cpue)*q_rate)*q_rate_fcn_HBD(styr_HBD_cpue);  //linear
        }
        if (iyear>2003) {q_rate_fcn_HBD(iyear)=q_rate_fcn_HBD(iyear-1);}
    }
 } //end q_rate conditional


 L_hb_bias=set_L_hb_bias;
 L_mrip_bias=set_L_mrip_bias;
 L_comm_bias=set_L_comm_bias;

 w_L=set_w_L;
 w_D=set_w_D;

 w_lc_Mbft=set_w_lc_Mbft;
 w_lc_cL=set_w_lc_cL;
 w_lc_cP=set_w_lc_cP;
 w_lc_HB=set_w_lc_HB;
 w_lc_HB_D=set_w_lc_HB_D;
 w_lc_mrip=set_w_lc_mrip;

 w_ac_Mbft=set_w_ac_Mbft;
 w_ac_Mcvt=set_w_ac_Mcvt;
 w_ac_cL=set_w_ac_cL;
 w_ac_cP=set_w_ac_cP;
 w_ac_HB=set_w_ac_HB;
 w_ac_mrip=set_w_ac_mrip;

 w_I_Mcvt=set_w_I_Mcvt;
 w_I_Mbft=set_w_I_Mbft;
 w_I_cL=set_w_I_cL;
 w_I_HB=set_w_I_HB;
 w_I_HBD=set_w_I_HBD;

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

 log_avg_F_cL=set_log_avg_F_cL;
 log_avg_F_cP=set_log_avg_F_cP;
 log_avg_F_cT=set_log_avg_F_cT;
 log_avg_F_HB=set_log_avg_F_HB;
 log_avg_F_mrip=set_log_avg_F_mrip;
 F_init_ratio=set_F_init_ratio;


 log_avg_F_comm_D=set_log_avg_F_comm_D;
 log_avg_F_HB_D=set_log_avg_F_HB_D;
 log_avg_F_mrip_D=set_log_avg_F_mrip_D;

 len_cv_val=set_len_cv;

 log_RO=set_log_R0;

 selpar_L50_Mbft= set_selpar_L50_Mbft;
 selpar_slope_Mbft=set_selpar_slope_Mbft;

 selpar_L50_Mcvt=set_selpar_L50_Mcvt;
 selpar_slope_Mcvt=set_selpar_slope_Mcvt;

 selpar_L50_cL2=set_selpar_L50_cL2;
 selpar_slope_cL2=set_selpar_slope_cL2;
 selpar_L50_cL3=set_selpar_L50_cL3;
 selpar_slope_cL3=set_selpar_slope_cL3;

 selpar_L50_cP2=set_selpar_L50_cP2;
 selpar_slope_cP2=set_selpar_slope_cP2;
 selpar_L50_cP3=set_selpar_L50_cP3;
 selpar_slope_cP3=set_selpar_slope_cP3;

 selpar_L50_HB1=set_selpar_L50_HB1;
 selpar_slope_HB1=set_selpar_slope_HB1;
 selpar_L50_HB2=set_selpar_L50_HB2;
 selpar_slope_HB2=set_selpar_slope_HB2;
 selpar_L50_HB3=set_selpar_L50_HB3;
 selpar_slope_HB3=set_selpar_slope_HB3;
 selpar_L50_HB4=set_selpar_L50_HB4;
 selpar_slope_HB4=set_selpar_slope_HB4;

 selpar_L50_mrip1=set_selpar_L50_mrip1;
 selpar_slope_mrip1=set_selpar_slope_mrip1;
 selpar_L50_mrip2=set_selpar_L50_mrip2;
 selpar_slope_mrip2=set_selpar_slope_mrip2;
 selpar_L50_mrip3=set_selpar_L50_mrip3;
 selpar_slope_mrip3=set_selpar_slope_mrip3;
 selpar_L50_mrip4=set_selpar_L50_mrip4;
 selpar_slope_mrip4=set_selpar_slope_mrip4;


 selpar_Age0_HB_D_logit=set_selpar_Age0_HB_D_logit;
 selpar_Age1_HB_D_logit=selpar_Age1_HB_D_logit;
 selpar_Age2_HB_D_logit=selpar_Age2_HB_D_logit;

SSB_msy_out=0.0;

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

```
 maturity_f=maturity_f_obs;
 prop_f=prop_f_obs;

 p_lenc_cL2=set_p_lenc_cL2;
 p_lenc_cL3=set_p_lenc_cL3;
 p_lenc_cP2=set_p_lenc_cP2;
 p_lenc_cP3=set_p_lenc_cP3;
 p_lenc_cT2=set_p_lenc_cT2;
 p_lenc_cT3=set_p_lenc_cT3;
 p_lenc_HB2=set_p_lenc_HB2;
 p_lenc_HB3=set_p_lenc_HB3;
 p_lenc_HB4=set_p_lenc_HB4;
 p_lenc_mrip2=set_p_lenc_mrip2;
 p_lenc_mrip3=set_p_lenc_mrip3;
 p_lenc_mrip4=set_p_lenc_mrip4;

 p_lenc_comm_D2=set_p_lenc_comm_D2;
 p_lenc_comm_D3=set_p_lenc_comm_D3;
 p_lenc_HB_D2=set_p_lenc_HB_D2;
 p_lenc_HB_D3=set_p_lenc_HB_D3;
 p_lenc_HB_D4=set_p_lenc_HB_D4;
 p_lenc_mrip_D1=set_p_lenc_mrip_D1;
 p_lenc_mrip_D2=set_p_lenc_mrip_D2;
 p_lenc_mrip_D3=set_p_lenc_mrip_D3;
 p_lenc_mrip_D4=set_p_lenc_mrip_D4;

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

 //multiplicative bias for early rec data
 obs_HB_L_wbias(styr_HB_L,endyr_L_HB_bias)=L_hb_bias*obs_HB_L(styr_HB_L,endyr_L_HB_bias);
 obs_HB_L_wbias((endyr_L_HB_bias+1),endyr_HB_L)=obs_HB_L((endyr_L_HB_bias+1),endyr_HB_L);

 obs_mrip_L_wbias(styr_mrip_L,endyr_L_mrip_bias)=L_hb_bias*obs_mrip_L(styr_mrip_L,endyr_L_mrip_bias);
 obs_mrip_L_wbias((endyr_L_mrip_bias+1),endyr_mrip_L)=obs_mrip_L((endyr_L_mrip_bias+1),endyr_mrip_L);


//Fill in sample sizes of comps, possibly sampled in nonconsec yrs
//Used primarily for output in R object
  nsamp_Mbft_lenc_allyr=missing;//"missing" defined in admb2r.cpp
  nsamp_cL_lenc_allyr=missing;
  nsamp_cP_lenc_allyr=missing;
  nsamp_HB_lenc_allyr=missing;
  nsamp_HB_D_lenc_allyr=missing;
  nsamp_mrip_lenc_allyr=missing;
  nsamp_Mbft_agec_allyr=missing;
  nsamp_Mcvt_agec_allyr=missing;
  nsamp_cL_agec_allyr=missing;
  nsamp_cP_agec_allyr=missing;
  nsamp_HB_agec_allyr=missing;
  nsamp_mrip_agec_allyr=missing;

  nfish_Mbft_lenc_allyr=missing;//"missing" defined in admb2r.cpp
  nfish_cL_lenc_allyr=missing;
  nfish_cP_lenc_allyr=missing;
  nfish_HB_lenc_allyr=missing;
  nfish_HB_D_lenc_allyr=missing;
  nfish_mrip_lenc_allyr=missing;
  nfish_Mbft_agec_allyr=missing;
  nfish_Mcvt_agec_allyr=missing;
  nfish_cL_agec_allyr=missing;
  nfish_cP_agec_allyr=missing;
  nfish_HB_agec_allyr=missing;
  nfish_mrip_agec_allyr=missing;


      for (iyear=1; iyear<=nyr_Mbft_lenc; iyear++)
         {  if (nsamp_Mbft_lenc(iyear)>maxSS_lenc)
             {nsamp_Mbft_lenc(iyear)=maxSS_lenc;}
            if (nsamp_Mbft_lenc(iyear)>=minSS_lenc)
             {nsamp_Mbft_lenc_allyr(yrs_Mbft_lenc(iyear))=nsamp_Mbft_lenc(iyear);
              nfish_Mbft_lenc_allyr(yrs_Mbft_lenc(iyear))=nfish_Mbft_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cL_lenc; iyear++)
         {  if (nsamp_cL_lenc(iyear)>maxSS_lenc)
             {nsamp_cL_lenc(iyear)=maxSS_lenc;}
            if (nsamp_cL_lenc(iyear)>=minSS_lenc)
             {nsamp_cL_lenc_allyr(yrs_cL_lenc(iyear))=nsamp_cL_lenc(iyear);
              nfish_cL_lenc_allyr(yrs_cL_lenc(iyear))=nfish_cL_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cP_lenc; iyear++)
         {  if (nsamp_cP_lenc(iyear)>maxSS_lenc)
             {nsamp_cP_lenc(iyear)=maxSS_lenc;}
            if (nsamp_cP_lenc(iyear)>=minSS_lenc)
             {nsamp_cP_lenc_allyr(yrs_cP_lenc(iyear))=nsamp_cP_lenc(iyear);
              nfish_cP_lenc_allyr(yrs_cP_lenc(iyear))=nfish_cP_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
         {  if (nsamp_HB_lenc(iyear)>maxSS_lenc)
             {nsamp_HB_lenc(iyear)=maxSS_lenc;}
            if (nsamp_HB_lenc(iyear)>=minSS_lenc)
             {nsamp_HB_lenc_allyr(yrs_HB_lenc(iyear))=nsamp_HB_lenc(iyear);
              nfish_HB_lenc_allyr(yrs_HB_lenc(iyear))=nfish_HB_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_HB_D_lenc; iyear++)
         {  if (nsamp_HB_D_lenc(iyear)>maxSS_lenc)
             {nsamp_HB_D_lenc(iyear)=maxSS_lenc;}
            if (nsamp_HB_D_lenc(iyear)>=minSS_lenc)
             {nsamp_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=nsamp_HB_D_lenc(iyear);
              nfish_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=nfish_HB_D_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_mrip_lenc; iyear++)
         {  if (nsamp_mrip_lenc(iyear)>maxSS_lenc)
             {nsamp_mrip_lenc(iyear)=maxSS_lenc;}
```

```
            if (nsamp_mrip_lenc(iyear)>=minSS_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_Mbft_agec; iyear++)
         {  if (nsamp_Mbft_agec(iyear)>maxSS_agec)
             {nsamp_Mbft_agec(iyear)=maxSS_agec;}
            if (nsamp_Mbft_agec(iyear)>=minSS_agec)
             {nsamp_Mbft_agec_allyr(yrs_Mbft_agec(iyear))=nsamp_Mbft_agec(iyear);
              nfish_Mbft_agec_allyr(yrs_Mbft_agec(iyear))=nfish_Mbft_agec(iyear);}}

      for (iyear=1; iyear<=nyr_Mcvt_agec; iyear++)
         {  if (nsamp_Mcvt_agec(iyear)>maxSS_agec)
             {nsamp_Mcvt_agec(iyear)=maxSS_agec;}
            if (nsamp_Mcvt_agec(iyear)>=minSS_agec)
             {nsamp_Mcvt_agec_allyr(yrs_Mcvt_agec(iyear))=nsamp_Mcvt_agec(iyear);
              nfish_Mcvt_agec_allyr(yrs_Mcvt_agec(iyear))=nfish_Mcvt_agec(iyear);}}

      for (iyear=1; iyear<=nyr_cL_agec; iyear++)
         {  if (nsamp_cL_agec(iyear)>maxSS_agec)
             {nsamp_cL_agec(iyear)=maxSS_agec;}
            if (nsamp_cL_agec(iyear)>=minSS_agec)
             {nsamp_cL_agec_allyr(yrs_cL_agec(iyear))=nsamp_cL_agec(iyear);
              nfish_cL_agec_allyr(yrs_cL_agec(iyear))=nfish_cL_agec(iyear);}}

      for (iyear=1; iyear<=nyr_cP_agec; iyear++)
         {  if (nsamp_cP_agec(iyear)>maxSS_agec)
             {nsamp_cP_agec(iyear)=maxSS_agec;}
            if (nsamp_cP_agec(iyear)>=minSS_agec)
             {nsamp_cP_agec_allyr(yrs_cP_agec(iyear))=nsamp_cP_agec(iyear);
              nfish_cP_agec_allyr(yrs_cP_agec(iyear))=nfish_cP_agec(iyear);}}

      for (iyear=1; iyear<=nyr_HB_agec; iyear++)
         {  if (nsamp_HB_agec(iyear)>maxSS_agec)
             {nsamp_HB_agec(iyear)=maxSS_agec;}
            if (nsamp_HB_agec(iyear)>=minSS_agec)
             {nsamp_HB_agec_allyr(yrs_HB_agec(iyear))=nsamp_HB_agec(iyear);
              nfish_HB_agec_allyr(yrs_HB_agec(iyear))=nfish_HB_agec(iyear);}}

      for (iyear=1; iyear<=nyr_mrip_agec; iyear++)
         {  if (nsamp_mrip_agec(iyear)>maxSS_agec)
             {nsamp_mrip_agec(iyear)=maxSS_agec;}
            if (nsamp_mrip_agec(iyear)>=minSS_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);}}

//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_cL.initialize(); F_cP.initialize(); F_cT.initialize(); F_HB.initialize(); F_mrip.initialize();
  F_comm_D.initialize(); F_HB_D.initialize(); F_mrip_D.initialize();

  L_cL_num.initialize(); L_cP_num.initialize(); L_cT_num.initialize(); L_HB_num.initialize(); L_mrip_num.initialize();
  D_comm_num.initialize(); D_HB_num.initialize(); D_mrip_num.initialize();

  F_cL_out.initialize(); F_cP_out.initialize(); F_cT_out.initialize(); F_HB_out.initialize(); F_mrip_out.initialize();

  F_comm_D_out.initialize(); F_HB_D_out.initialize(); F_mrip_D_out.initialize();

  pred_cL_L_klb.initialize();pred_cP_L_klb.initialize();pred_cT_L_klb.initialize(); pred_HB_L_klb.initialize(); pred_mrip_L_klb.initialize();
  pred_cL_L_knum.initialize();pred_cP_L_knum.initialize();pred_cT_L_knum.initialize(); pred_HB_L_knum.initialize(); pred_mrip_L_knum.initialize();

  pred_comm_D_klb.initialize(); pred_HB_D_klb.initialize(); pred_mrip_D_klb.initialize();
  pred_comm_D_knum.initialize(); pred_HB_D_knum.initialize(); pred_mrip_D_knum.initialize();

  sel_Mcvt.initialize(); sel_Mbft.initialize();
  sel_cL.initialize(); sel_cP.initialize(); sel_cT.initialize();
  sel_HB.initialize(); sel_mrip.initialize();
  sel_comm_D.initialize(); sel_HB_D.initialize(); sel_mrip_D.initialize();

  log_rec_dev_output.initialize();
  log_Nage_dev_output.initialize();
  log_rec_dev.initialize();
  log_Nage_dev.initialize();


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
TOP_OF_MAIN_SECTION
  arrmblsize=20000000;
  gradient_structure::set_MAX_NVAR_OFFSET(1600);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(2000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(2000000);
  gradient_structure::set_NUM_DEPENDENT_VARIABLES(500);


//>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
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();
 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_dead_discards();
 //cout << "got discards" << 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_M_at_age
//    Mscale_len=Linf*(1.0-mfexp(-K*(Mscale_ages-t0+0.5)));
//    Mscale_wgt_g=wgtpar_a*pow(Mscale_len,wgtpar_b);
//    M_lorenzen=3.69*pow(Mscale_wgt_g,-0.305);
//    cum_surv_1plus=mfexp(-max_obs_age*M_constant);
//    M=M_lorenzen(1,nages)*(-log(cum_surv_1plus)/sum(M_lorenzen(1,max_obs_age)));


FUNCTION get_length_weight_at_age
  //compute mean length (mm) and weight (whole) at age
    meanlen_TL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));    //total length in mm
    wgt_g=wgtpar_a*pow(meanlen_TL,wgtpar_b);             //wgt in grams
    wgt_kg=g2kg*wgt_g;                                   //wgt in kilograms
    wgt_mt=g2mt*wgt_g;                                   //mt of whole wgt: g2mt converts g to mt
    wgt_klb=mt2klb*wgt_mt;                               //1000 lb of whole wgt
    wgt_lb=mt2lb*wgt_mt;                                 //1000 lb of whole wgt
    fecundity=fecpar_batches*mfexp(fecpar_a+wgt_g*fecpar_b)/fecpar_scale;    //fecundity at age, scaled

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

FUNCTION get_length_at_age_dist

//compute matrix of length at age, based on the normal distribution
  len_cv=len_cv_val;
  len_sd=elem_prod(len_cv, meanlen_TL);
  for (iage=1;iage<=nages;iage++)
  {
    //len_cv(iage)=mfexp(log_len_cv+log_len_cv_dev(iage)); len_sd(iage)=len_cv(iage)*meanlen_TL(iage);
    for (ilen=1;ilen<=nlenbins_all;ilen++)
     { lenprob_all(iage,ilen)=(mfexp(-(square(lenbins_all(ilen)-meanlen_TL(iage))/
       (2.*square(len_sd(iage)))))/(sqrt2pi*len_sd(iage)));
     }
    lenprob_all(iage)/=sum(lenprob_all(iage)); //standardize to approximate integration and to account for truncated normal (i.e., no sizes<smallest)
    for (ilen=1;ilen<=nlenbins;ilen++) {lenprob(iage,ilen)=lenprob_all(iage,ilen);}
    for (ilen=nlenbins+1;ilen<=nlenbins_all;ilen++){lenprob(iage)(nlenbins)=lenprob(iage)(nlenbins)+lenprob_all(iage)(ilen);} //plus group
  }



  //fishery/fleet specific length probs, assumed normal prior to size limits
  lenprob_Mbft=lenprob;

  lenprob_cL1=lenprob;
  lenprob_cL2_all=lenprob_all;    //values may be adjusted based on size limit
  lenprob_cL3_all=lenprob_all;    //values may be adjusted based on size limit

  lenprob_cP1=lenprob;
  lenprob_cP2_all=lenprob_all;    //values may be adjusted based on size limit
  lenprob_cP3_all=lenprob_all;    //values may be adjusted based on size limit

  lenprob_cT1=lenprob;
  lenprob_cT2_all=lenprob_all;    //values may be adjusted based on size limit

  lenprob_HB1=lenprob;
  lenprob_HB2_all=lenprob_all;    //values may be adjusted based on size limit
  lenprob_HB3_all=lenprob_all;    //values may be adjusted based on size limit
  lenprob_HB4_all=lenprob_all;    //values may be adjusted based on size limit

  lenprob_mrip1=lenprob;
  lenprob_mrip2_all=lenprob_all;   //values may be adjusted based on size limit
  lenprob_mrip3_all=lenprob_all;   //values may be adjusted based on size limit
  lenprob_mrip4_all=lenprob_all;   //values may be adjusted based on size limit

  lenprob_comm_D2_all=lenprob_all; //values may be adjusted based on size limit
  lenprob_comm_D3_all=lenprob_all; //values may be adjusted based on size limit
  lenprob_HB_D2_all=lenprob_all;   //values may be adjusted based on size limit
  lenprob_HB_D3_all=lenprob_all;   //values may be adjusted based on size limit
  lenprob_HB_D4_all=lenprob_all;   //values may be adjusted based on size limit
```

```
lenprob_mrip_D1_all=lenprob_all; //values may be adjusted based on size limit
lenprob_mrip_D2_all=lenprob_all; //values may be adjusted based on size limit
lenprob_mrip_D3_all=lenprob_all; //values may be adjusted based on size limit
lenprob_mrip_D4_all=lenprob_all; //values may be adjusted based on size limit

for (iage=1;iage<=nages;iage++)
{
  for (ilen=1;ilen<=nlenbins_all;ilen++)
  {
      if (lenbins_all(ilen) < limit_8in)  //Landings block two
      {
        lenprob_cL2_all(iage,ilen)=p_lenc_cL2*lenprob_all(iage,ilen);
        lenprob_cP2_all(iage,ilen)=p_lenc_cP2*lenprob_all(iage,ilen);
        lenprob_cT2_all(iage,ilen)=p_lenc_cT2*lenprob_all(iage,ilen);
        lenprob_HB2_all(iage,ilen)=p_lenc_HB2*lenprob_all(iage,ilen);
        lenprob_mrip2_all(iage,ilen)=p_lenc_mrip2*lenprob_all(iage,ilen);

      }
      if (lenbins_all(ilen) < limit_10in) //Landings block three
      {
        lenprob_cL3_all(iage,ilen)=p_lenc_cL3*lenprob_all(iage,ilen);
        lenprob_cP3_all(iage,ilen)=p_lenc_cP3*lenprob_all(iage,ilen);
        lenprob_HB3_all(iage,ilen)=p_lenc_HB3*lenprob_all(iage,ilen);
        lenprob_mrip3_all(iage,ilen)=p_lenc_mrip3*lenprob_all(iage,ilen);
       }
      if (lenbins_all(ilen) < limit_12in) //Landings block four
      {
        lenprob_HB4_all(iage,ilen)=p_lenc_HB4*lenprob_all(iage,ilen);
        lenprob_mrip4_all(iage,ilen)=p_lenc_mrip4*lenprob_all(iage,ilen);
       }


      if (lenbins_all(ilen) > limit_disc) //Discards block 1
      { lenprob_mrip_D1_all(iage,ilen)=p_lenc_mrip_D1*lenprob_all(iage,ilen);}

      if (lenbins_all(ilen) > limit_8in) //Discards block two
      {
        lenprob_comm_D2_all(iage,ilen)=p_lenc_comm_D2*lenprob_all(iage,ilen);
        lenprob_HB_D2_all(iage,ilen)=p_lenc_HB_D2*lenprob_all(iage,ilen);
        lenprob_mrip_D2_all(iage,ilen)=p_lenc_mrip_D2*lenprob_all(iage,ilen);
      }

      if (lenbins_all(ilen) > limit_10in) //Discards block three
      {
        lenprob_comm_D3_all(iage,ilen)=p_lenc_comm_D3*lenprob_all(iage,ilen);
        lenprob_HB_D3_all(iage,ilen)=p_lenc_HB_D3*lenprob_all(iage,ilen);
        lenprob_mrip_D3_all(iage,ilen)=p_lenc_mrip_D3*lenprob_all(iage,ilen);
      }

      if (lenbins_all(ilen) > limit_12in) //Discards block three
      {
        lenprob_HB_D4_all(iage,ilen)=p_lenc_HB_D4*lenprob_all(iage,ilen);
        lenprob_mrip_D4_all(iage,ilen)=p_lenc_mrip_D4*lenprob_all(iage,ilen);
      }


  }  //end ilen loop


  if (iage>=4)  //compute prior to standardizing
      {vecprob_HB_D2(iage)=sum(lenprob_HB_D2_all(iage));
       vecprob_HB_D3(iage)=sum(lenprob_HB_D3_all(iage));
       vecprob_HB_D4(iage)=sum(lenprob_HB_D4_all(iage));
      }

  lenprob_cL2_all(iage)/=sum(lenprob_cL2_all(iage));       //standardize
  lenprob_cL3_all(iage)/=sum(lenprob_cL3_all(iage));       //standardize
  lenprob_cP2_all(iage)/=sum(lenprob_cP2_all(iage));       //standardize
  lenprob_cP3_all(iage)/=sum(lenprob_cP3_all(iage));       //standardize
  lenprob_cT2_all(iage)/=sum(lenprob_cT2_all(iage));       //standardize
  lenprob_HB2_all(iage)/=sum(lenprob_HB2_all(iage));       //standardize
  lenprob_HB3_all(iage)/=sum(lenprob_HB3_all(iage));       //standardize
  lenprob_HB4_all(iage)/=sum(lenprob_HB4_all(iage));       //standardize
  lenprob_mrip2_all(iage)/=sum(lenprob_mrip2_all(iage));   //standardize
  lenprob_mrip3_all(iage)/=sum(lenprob_mrip3_all(iage));   //standardize
  lenprob_mrip4_all(iage)/=sum(lenprob_mrip4_all(iage));   //standardize

  lenprob_comm_D2_all(iage)/=sum(lenprob_comm_D2_all(iage));   //standardize
  lenprob_comm_D3_all(iage)/=sum(lenprob_comm_D3_all(iage));   //standardize
  lenprob_HB_D2_all(iage)/=sum(lenprob_HB_D2_all(iage));       //standardize
  lenprob_HB_D3_all(iage)/=sum(lenprob_HB_D3_all(iage));       //standardize
  lenprob_HB_D4_all(iage)/=sum(lenprob_HB_D4_all(iage));       //standardize
  lenprob_mrip_D2_all(iage)/=sum(lenprob_mrip_D2_all(iage));   //standardize
  lenprob_mrip_D3_all(iage)/=sum(lenprob_mrip_D3_all(iage));   //standardize
  lenprob_mrip_D4_all(iage)/=sum(lenprob_mrip_D4_all(iage));   //standardize

  for (ilen=1;ilen<=nlenbins;ilen++)
    {lenprob_cL2(iage,ilen)=lenprob_cL2_all(iage,ilen);
     lenprob_cL3(iage,ilen)=lenprob_cL3_all(iage,ilen);
     lenprob_cP2(iage,ilen)=lenprob_cP2_all(iage,ilen);
     lenprob_cP3(iage,ilen)=lenprob_cP3_all(iage,ilen);
     lenprob_cT2(iage,ilen)=lenprob_cT2_all(iage,ilen);
     lenprob_HB2(iage,ilen)=lenprob_HB2_all(iage,ilen);
     lenprob_HB3(iage,ilen)=lenprob_HB3_all(iage,ilen);
     lenprob_HB4(iage,ilen)=lenprob_HB4_all(iage,ilen);
     lenprob_mrip2(iage,ilen)=lenprob_mrip2_all(iage,ilen);
     lenprob_mrip3(iage,ilen)=lenprob_mrip3_all(iage,ilen);
     lenprob_mrip4(iage,ilen)=lenprob_mrip4_all(iage,ilen);
     lenprob_comm_D2(iage,ilen)=lenprob_comm_D2_all(iage,ilen);
     lenprob_comm_D3(iage,ilen)=lenprob_comm_D3_all(iage,ilen);
     lenprob_HB_D2(iage,ilen)=lenprob_HB_D2_all(iage,ilen);
     lenprob_HB_D3(iage,ilen)=lenprob_HB_D3_all(iage,ilen);
     lenprob_HB_D4(iage,ilen)=lenprob_HB_D4_all(iage,ilen);
     lenprob_mrip_D2(iage,ilen)=lenprob_mrip_D2_all(iage,ilen);
```

```
       lenprob_mrip_D3(iage,ilen)=lenprob_mrip_D3_all(iage,ilen);
       lenprob_mrip_D4(iage,ilen)=lenprob_mrip_D4_all(iage,ilen);
      }

    for (ilen=nlenbins+1;ilen<=nlenbins_all;ilen++) //plus group
      {
      lenprob_cL2(iage)(nlenbins)=lenprob_cL2(iage)(nlenbins)+lenprob_cL2_all(iage)(ilen);
      lenprob_cL3(iage)(nlenbins)=lenprob_cL3(iage)(nlenbins)+lenprob_cL3_all(iage)(ilen);
      lenprob_cP2(iage)(nlenbins)=lenprob_cP2(iage)(nlenbins)+lenprob_cP2_all(iage)(ilen);
      lenprob_cP3(iage)(nlenbins)=lenprob_cP3(iage)(nlenbins)+lenprob_cP3_all(iage)(ilen);
      lenprob_cT2(iage)(nlenbins)=lenprob_cT2(iage)(nlenbins)+lenprob_cT2_all(iage)(ilen);
      lenprob_HB2(iage)(nlenbins)=lenprob_HB2(iage)(nlenbins)+lenprob_HB2_all(iage)(ilen);
      lenprob_HB3(iage)(nlenbins)=lenprob_HB3(iage)(nlenbins)+lenprob_HB3_all(iage)(ilen);
      lenprob_HB4(iage)(nlenbins)=lenprob_HB4(iage)(nlenbins)+lenprob_HB4_all(iage)(ilen);
      lenprob_mrip2(iage)(nlenbins)=lenprob_mrip2(iage)(nlenbins)+lenprob_mrip2_all(iage)(ilen);
      lenprob_mrip3(iage)(nlenbins)=lenprob_mrip3(iage)(nlenbins)+lenprob_mrip3_all(iage)(ilen);
      lenprob_mrip4(iage)(nlenbins)=lenprob_mrip4(iage)(nlenbins)+lenprob_mrip4_all(iage)(ilen);
      lenprob_comm_D2(iage)(nlenbins)=lenprob_comm_D2(iage)(nlenbins)+lenprob_comm_D2_all(iage)(ilen);
      lenprob_comm_D3(iage)(nlenbins)=lenprob_comm_D3(iage)(nlenbins)+lenprob_comm_D3_all(iage)(ilen);
      lenprob_HB_D2(iage)(nlenbins)=lenprob_HB_D2(iage)(nlenbins)+lenprob_HB_D2_all(iage)(ilen);
      lenprob_HB_D3(iage)(nlenbins)=lenprob_HB_D3(iage)(nlenbins)+lenprob_HB_D3_all(iage)(ilen);
      lenprob_HB_D4(iage)(nlenbins)=lenprob_HB_D4(iage)(nlenbins)+lenprob_HB_D4_all(iage)(ilen);
      lenprob_mrip_D2(iage)(nlenbins)=lenprob_mrip_D2(iage)(nlenbins)+lenprob_mrip_D2_all(iage)(ilen);
      lenprob_mrip_D3(iage)(nlenbins)=lenprob_mrip_D3(iage)(nlenbins)+lenprob_mrip_D3_all(iage)(ilen);
      lenprob_mrip_D4(iage)(nlenbins)=lenprob_mrip_D4(iage)(nlenbins)+lenprob_mrip_D4_all(iage)(ilen);
      }

  } //end iage loop

FUNCTION get_weight_at_age_landings
  //fleets under identical size limits are set equal at end of fcn
  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
    len_cL_mm(iyear)=meanlen_TL;
    wgt_cL_klb(iyear)=wgt_klb;
    //len_cP_mm(iyear)=meanlen_TL;
    //wgt_cP_klb(iyear)=wgt_klb;
    //len_cT_mm(iyear)=meanlen_TL;
    //wgt_cT_klb(iyear)=wgt_klb;

    //len_HB_mm(iyear)=meanlen_TL;
    //wgt_HB_klb(iyear)=wgt_klb;
    len_mrip_mm(iyear)=meanlen_TL;
    wgt_mrip_klb(iyear)=wgt_klb;

    for (iage=1;iage<=nages; iage++)
    {
    len_mrip_D_mm(iyear,iage)=sum(elem_prod(lenprob_mrip_D2_all(iage),lenbins_all)); //assumes same size distn in period 1 as in period 2
    }
    wgt_mrip_D_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_D_mm(iyear),wgtpar_b);
  } // end iyear loop


  for (iyear=(endyr_period1+1); iyear<=endyr_period2; iyear++)
  {
    for (iage=1;iage<=nages; iage++)
    {
    len_cL_mm(iyear,iage)=sum(elem_prod(lenprob_cL2_all(iage),lenbins_all));
    //len_cP_mm(iyear,iage)=sum(elem_prod(lenprob_cP2_all(iage),lenbins_all));
    //len_cT_mm(iyear,iage)=sum(elem_prod(lenprob_cT2_all(iage),lenbins_all));
    //len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB2_all(iage),lenbins_all));
    len_mrip_mm(iyear,iage)=sum(elem_prod(lenprob_mrip2_all(iage),lenbins_all));
    len_comm_D_mm(iyear,iage)=sum(elem_prod(lenprob_comm_D2_all(iage),lenbins_all));
    //len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D2_all(iage),lenbins_all));
    len_mrip_D_mm(iyear,iage)=sum(elem_prod(lenprob_mrip_D2_all(iage),lenbins_all));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    //wgt_cP_klb(iyear)=g2klb*wgtpar_a*pow(len_cP_mm(iyear),wgtpar_b);
    //wgt_cT_klb(iyear)=g2klb*wgtpar_a*pow(len_cT_mm(iyear),wgtpar_b);
    //wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_mrip_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_mm(iyear),wgtpar_b);
    wgt_comm_D_klb(iyear)=g2klb*wgtpar_a*pow(len_comm_D_mm(iyear),wgtpar_b);
    //wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
    wgt_mrip_D_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_D_mm(iyear),wgtpar_b);
  }

  for (iyear=(endyr_period2+1); iyear<=endyr; iyear++) //comm only
  {
    for (iage=1;iage<=nages; iage++)
    {
    len_cL_mm(iyear,iage)=sum(elem_prod(lenprob_cL3_all(iage),lenbins_all));
    //len_cP_mm(iyear,iage)=sum(elem_prod(lenprob_cP3_all(iage),lenbins_all));
    len_comm_D_mm(iyear,iage)=sum(elem_prod(lenprob_comm_D3_all(iage),lenbins_all));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    //wgt_cP_klb(iyear)=g2klb*wgtpar_a*pow(len_cP_mm(iyear),wgtpar_b);
    wgt_comm_D_klb(iyear)=g2klb*wgtpar_a*pow(len_comm_D_mm(iyear),wgtpar_b);
  }

  for (iyear=(endyr_period2+1); iyear<=endyr_recr_period3; iyear++) //rec only
  {
    for (iage=1;iage<=nages; iage++)
    {
    //len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB3_all(iage),lenbins_all));
    len_mrip_mm(iyear,iage)=sum(elem_prod(lenprob_mrip3_all(iage),lenbins_all));
    //len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D3_all(iage),lenbins_all));
    len_mrip_D_mm(iyear,iage)=sum(elem_prod(lenprob_mrip_D3_all(iage),lenbins_all));
    }
    //wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_mrip_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_mm(iyear),wgtpar_b);
    //wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
    wgt_mrip_D_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_D_mm(iyear),wgtpar_b);
  }
```

```
  for (iyear=(endyr_recr_period3+1); iyear<=endyr; iyear++) //rec only
  {
    for (iage=1;iage<=nages; iage++)
    {
    //len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB4_all(iage),lenbins_all));
    len_mrip_mm(iyear,iage)=sum(elem_prod(lenprob_mrip4_all(iage),lenbins_all));
    //len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D4_all(iage),lenbins_all));
    len_mrip_D_mm(iyear,iage)=sum(elem_prod(lenprob_mrip_D4_all(iage),lenbins_all));
    }
    //wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_mrip_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_mm(iyear),wgtpar_b);
    //wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
    wgt_mrip_D_klb(iyear)=g2klb*wgtpar_a*pow(len_mrip_D_mm(iyear),wgtpar_b);
  }

    //identical fleets set equal here (for speed)
    len_cP_mm=len_cL_mm;  wgt_cP_klb=wgt_cL_klb;
    len_cT_mm=len_cL_mm;  wgt_cT_klb=wgt_cL_klb;
    len_HB_mm=len_mrip_mm;  wgt_HB_klb=wgt_mrip_klb;
    len_HB_D_mm=len_mrip_D_mm;  wgt_HB_D_klb=wgt_mrip_D_klb;

  for (iyear=styr_comm_closed; iyear<=endyr; iyear++) //overwrite last two yrs comm discards, accnt for quotas
  { len_comm_D_mm(iyear)=Dprop_comm_sel_D*len_comm_D_mm(iyear)+Dprop_comm_sel_cL*len_cL_mm(iyear)+
                         Dprop_comm_sel_cP*len_cP_mm(iyear);
    wgt_comm_D_klb(iyear)=g2klb*wgtpar_a*pow(len_comm_D_mm(iyear),wgtpar_b);
  }

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

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


FUNCTION get_selectivity

// ------- compute landings selectivities by period

  //---flat-topped sels---------------------------
  sel_Mbft_vec=logistic(agebins, selpar_L50_Mbft, selpar_slope_Mbft);
  sel_Mcvt_vec=logistic(agebins, selpar_L50_Mcvt, selpar_slope_Mcvt);

  sel_cL_2=logistic(agebins, selpar_L50_cL2, selpar_slope_cL2);
  sel_cL_3=logistic(agebins, selpar_L50_cL3, selpar_slope_cL3);

  sel_cP_2=logistic(agebins, selpar_L50_cP2, selpar_slope_cP2);
  sel_cP_3=logistic(agebins, selpar_L50_cP3, selpar_slope_cP3);

  sel_HB_1=logistic(agebins, selpar_L50_HB1, selpar_slope_HB1);
  sel_HB_2=logistic(agebins, selpar_L50_HB2, selpar_slope_HB2);
  sel_HB_3=logistic(agebins, selpar_L50_HB3, selpar_slope_HB3);
  sel_HB_4=logistic(agebins, selpar_L50_HB4, selpar_slope_HB4);

  selpar_L50_mrip1=selpar_L50_HB1;
  selpar_slope_mrip1=selpar_slope_HB1;
  selpar_L50_mrip2=selpar_L50_HB2;
  selpar_slope_mrip2=selpar_slope_HB2;
  selpar_L50_mrip3=selpar_L50_HB3;
  selpar_slope_mrip3=selpar_slope_HB3;
  selpar_L50_mrip4=selpar_L50_HB4;
  selpar_slope_mrip4=selpar_slope_HB4;

  sel_mrip_1=logistic(agebins, selpar_L50_mrip1, selpar_slope_mrip1);
  sel_mrip_2=logistic(agebins, selpar_L50_mrip2, selpar_slope_mrip2);
  sel_mrip_3=logistic(agebins, selpar_L50_mrip3, selpar_slope_mrip3);
  sel_mrip_4=logistic(agebins, selpar_L50_mrip4, selpar_slope_mrip4);


//-----------fill in years-------------------------------------------

  //Period 1:
  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
     sel_Mbft(iyear)=sel_Mbft_vec;
     sel_Mcvt(iyear)=sel_Mcvt_vec;
     sel_cL(iyear)=sel_cL_2; //commercial handline sel mirrors period 2
     sel_cP(iyear)=sel_cP_2; //commercial handline sel mirrors period 2
     sel_HB(iyear)=sel_HB_1;
     sel_mrip(iyear)=sel_mrip_1;
  }

  //Period 2:
  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  {
     sel_Mbft(iyear)=sel_Mbft_vec;
     sel_Mcvt(iyear)=sel_Mcvt_vec;
     sel_cL(iyear)=sel_cL_2;
     sel_cP(iyear)=sel_cP_2;
     sel_HB(iyear)=sel_HB_2;
     sel_mrip(iyear)=sel_mrip_2;
  }

  //Period 3
```

```
  for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
  {
     sel_Mbft(iyear)=sel_Mbft_vec;
     sel_Mcvt(iyear)=sel_Mcvt_vec;
     sel_cL(iyear)=sel_cL_3;
     sel_cP(iyear)=sel_cP_3;
     sel_HB(iyear)=sel_HB_3;
     sel_mrip(iyear)=sel_mrip_3;
  }
  //Period 4: rec only, overwrites last few yrs calculated for period 3
  for (iyear=endyr_recr_period3+1; iyear<=endyr; iyear++)
  {
     sel_HB(iyear)=sel_HB_4;
     sel_mrip(iyear)=sel_mrip_4;
  }

  //set selectivities that mirror others
  sel_cT=sel_cP;
  //sel_mrip=sel_HB;


//---Discard selectivities--------------------------------
//-------------------------------------------------------
  //mrip and comm mirror headboat discard selectivity

  selpar_Age0_HB_D=1.0/(1.0+mfexp(-selpar_Age0_HB_D_logit));
  selpar_Age1_HB_D=1.0/(1.0+mfexp(-selpar_Age1_HB_D_logit));
  selpar_Age2_HB_D=1.0/(1.0+mfexp(-selpar_Age2_HB_D_logit));


 //Assume same sel of age 0's, 1's across periods
  sel_HB_D_2(1)=selpar_Age0_HB_D; sel_HB_D_3(1)=selpar_Age0_HB_D; sel_HB_D_4(1)=selpar_Age0_HB_D;
  sel_HB_D_2(2)=selpar_Age1_HB_D; sel_HB_D_3(2)=selpar_Age1_HB_D; sel_HB_D_4(2)=selpar_Age1_HB_D;
  sel_HB_D_2(3)=selpar_Age2_HB_D; sel_HB_D_3(3)=selpar_Age2_HB_D; sel_HB_D_4(3)=selpar_Age2_HB_D;
  sel_HB_D_2(4)=1.0; sel_HB_D_3(4)=1.0; sel_HB_D_4(4)=1.0;

  for (iage=5; iage<=nages; iage++)
      {sel_HB_D_2(iage)=vecprob_HB_D2(iage);
       sel_HB_D_3(iage)=vecprob_HB_D3(iage);
       sel_HB_D_4(iage)=vecprob_HB_D4(iage);
       }

  //Period 1: assumed same as in period 1, no commercial discards
  for (iyear=styr; iyear<=endyr_period1; iyear++)
      {sel_HB_D(iyear)=sel_HB_D_2;}

  //Period 2:
  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
      {sel_HB_D(iyear)=sel_HB_D_2;
       sel_comm_D(iyear)=sel_HB_D_2;
      }

  //Period 3:
  for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
  {sel_HB_D(iyear)=sel_HB_D_3;
   sel_comm_D(iyear)=sel_HB_D_3;}

  //Period 3: discard quota: wghted average,, overwrites last few yrs calculated for period 3
  for (iyear=styr_comm_closed; iyear<=endyr; iyear++)
  {sel_comm_D(iyear)=Dprop_comm_sel_D*sel_HB_D_3 + Dprop_comm_sel_cL*sel_cL_3 +
                     Dprop_comm_sel_cP*sel_cP_3;
   sel_comm_D(iyear)=sel_comm_D(iyear)/max(sel_comm_D(iyear));}

  //Period 4: hb and mrip only, overwrites last few yrs calculated for period 3
  for (iyear=endyr_recr_period3+1; iyear<=endyr; iyear++)
  {sel_HB_D(iyear)=sel_HB_D_4;}

  //mrip discard selectivity same as headboat;
  sel_mrip_D=sel_HB_D;

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  //initialization F is avg from first 3 yrs of observed landings
  log_F_dev_init_cL=sum(log_F_dev_cL(styr_cL_L,(styr_cL_L+2)))/3.0;
  log_F_dev_init_cP=sum(log_F_dev_cL(styr_cP_L,(styr_cP_L+2)))/3.0;
  log_F_dev_init_cT=sum(log_F_dev_cL(styr_cT_L,(styr_cT_L+2)))/3.0;
  log_F_init_HB=sum(log_F_dev_HB(styr_HB_L,(styr_HB_L+2)))/3.0;
  log_F_dev_init_mrip=sum(log_F_dev_mrip(styr_mrip_L,(styr_mrip_L+2)))/3.0;
  log_F_dev_init_mrip_D=sum(log_F_dev_mrip_D(styr_mrip_D,(styr_mrip_D+2)))/3.0;

  log_F_dev_comm_D2=sum(log_F_dev_comm_D(styr_cL_D,(styr_cL_D+5)))/6.0; //for comm D 1984-1992

  //cout<<styr<<endl;
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    //-------------
    if(iyear>=styr_cL_L & iyear<=endyr_cL_L)
    {  F_cL_out(iyear)=mfexp(log_avg_F_cL+log_F_dev_cL(iyear)); //}
       //if (iyear<styr_cL_L){F_cL_out(iyear)=mfexp(log_avg_F_cL+log_F_dev_init_cL);}
       F_cL(iyear)=sel_cL(iyear)*F_cL_out(iyear);
       Fsum(iyear)+=F_cL_out(iyear);
    }

    //-------------
    if(iyear>=styr_cP_L & iyear<=endyr_cP_L)
    {  F_cP_out(iyear)=mfexp(log_avg_F_cP+log_F_dev_cP(iyear)); //}
       //if (iyear<styr_cP_L) {F_cP_out(iyear)=0.0;}
       F_cP(iyear)=sel_cP(iyear)*F_cP_out(iyear);
       Fsum(iyear)+=F_cP_out(iyear);
    }

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

```
    if(iyear>=styr_cT_L & iyear<=endyr_cT_L)
    {  F_cT_out(iyear)=mfexp(log_avg_F_cT+log_F_dev_cT(iyear)); //}
      // if (iyear<styr_cT_L) {F_cT_out(iyear)=0.0;}
       F_cT(iyear)=sel_cT(iyear)*F_cT_out(iyear);
       Fsum(iyear)+=F_cT_out(iyear);
    }

    //-------------
    if(iyear>=styr_HB_L & iyear<=endyr_HB_L)
    {  F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_dev_HB(iyear));//}
    // if (iyear<styr_HB_L){F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_init_HB);}
       F_HB(iyear)=sel_HB(iyear)*F_HB_out(iyear);
       Fsum(iyear)+=F_HB_out(iyear);
    }

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

    //discards-------------

    if(iyear>=styr_cL_D & iyear<=endyr_cL_D)
      {F_comm_D_out(iyear)=mfexp(log_avg_F_comm_D+log_F_dev_comm_D(iyear));}
    if(iyear > endyr_period1 & iyear < styr_cL_D)
      {F_comm_D_out(iyear)=mfexp(log_avg_F_comm_D+log_F_dev_comm_D2);}
    F_comm_D(iyear)=sel_comm_D(iyear)*F_comm_D_out(iyear);
    Fsum(iyear)+=F_comm_D_out(iyear);


    if(iyear>=styr_HB_D & iyear<=endyr_HB_D)
    {  F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_dev_HB_D(iyear));
       F_HB_D(iyear)=sel_HB_D(iyear)*F_HB_D_out(iyear);
       Fsum(iyear)+=F_HB_D_out(iyear);
    }
    if(iyear<styr_mrip_D)
       {F_mrip_D_out(iyear)=mfexp(log_avg_F_mrip_D+log_F_dev_init_mrip_D);}
    if(iyear>=styr_mrip_D& iyear<=endyr_mrip_D)
       { F_mrip_D_out(iyear)=mfexp(log_avg_F_mrip_D+log_F_dev_mrip_D(iyear));}
    F_mrip_D(iyear)=sel_mrip_D(iyear)*F_mrip_D_out(iyear);
    Fsum(iyear)+=F_mrip_D_out(iyear);

    //Total F at age
    F(iyear)=F_cL(iyear); //first in additive series (NO +=)
    F(iyear)+=F_cP(iyear);
    F(iyear)+=F_cT(iyear);
    F(iyear)+=F_HB(iyear);
    F(iyear)+=F_mrip(iyear);

    F(iyear)+=F_comm_D(iyear);
    F(iyear)+=F_HB_D(iyear);
    F(iyear)+=F_mrip_D(iyear);

    Fapex(iyear)=max(F(iyear));
    Z(iyear)=M+F(iyear);
  }  //end iyear


FUNCTION get_bias_corr
  //may exclude last BiasCor_exclude_yrs yrs bc constrained or lack info to estimate
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,(endyr-BiasCor_exclude_yrs))-
              sum(log_rec_dev(styr_rec_dev,(endyr-BiasCor_exclude_yrs)))
              /(nyrs_rec-BiasCor_exclude_yrs))/(nyrs_rec-BiasCor_exclude_yrs-1.0);
  //if (set_BiasCor <= 0.0) {BiasCor=mfexp(var_rec_dev/2.0);}   //bias correction
  rec_sigma_sq=square(rec_sigma);
  rec_sigma_sqd2=rec_sigma_sq/2.0;
  if (set_BiasCor <= 0.0) {BiasCor=mfexp(rec_sigma_sqd2);}   //bias correction
  else {BiasCor=set_BiasCor;}


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

  F_initial=sel_cL(styr)*mfexp(log_avg_F_cL+log_F_dev_init_cL)+
            sel_cP(styr)*mfexp(log_avg_F_cP+log_F_dev_init_cP)+
            sel_cT(styr)*mfexp(log_avg_F_cT+log_F_dev_init_cT)+
            sel_HB(styr)*mfexp(log_avg_F_HB+log_F_init_HB)+
            sel_mrip(styr)*mfexp(log_avg_F_mrip+log_F_dev_init_mrip)+
            sel_mrip_D(styr)*mfexp(log_avg_F_mrip_D+log_F_dev_init_mrip_D);
  Z_initial=M+F_init_ratio*F_initial;


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

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

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

```
                (BiasCor*4.0*steep*spr_initial-spr_F0*(1.0-steep));} //with bias correction

  if(R1<10.0) {R1=10.0;} //Avoid negative (or unreasonably low) 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));
  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)) //recruitment follows S-R curve exactly
    {
        N(iyear+1,1)=0.0;   //no age 0's mature in SSB calculations, value replaced below for abundance calcs
        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_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));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
        N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear+1));
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
    }

    else   //recruitment follows S-R curve with lognormal deviation
    {
        N(iyear+1,1)=0.0;   //no age 0's mature in SSB calculations, value replaced below for abundance calcs
        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_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));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
        N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear+1))*mfexp(log_rec_dev(iyear+1));
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year

    }
  } //end iyear

    ////last year (projection) cannot compute recuitment of age 0's past terminal yr bc spawning occurs at spawn_time_frac
    //N(endyr+1,1)=SR_func(R0, steep, spr_F0, SSB(endyr));
    //N(endyr+1)(2,nages)=++elem_prod(N(endyr)(1,nages-1),(mfexp(-1.*Z(endyr)(1,nages-1))));
    //N(endyr+1,nages)+=N(endyr,nages)*mfexp(-1.*Z(endyr,nages));//plus group
    //SSB(endyr+1)=sum(elem_prod(N(endyr+1),reprod));


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

FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_cL_num(iyear,iage)=N(iyear,iage)*F_cL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cP_num(iyear,iage)=N(iyear,iage)*F_cP(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cT_num(iyear,iage)=N(iyear,iage)*F_cT(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_HB_num(iyear,iage)=N(iyear,iage)*F_HB(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_mrip_num(iyear,iage)=N(iyear,iage)*F_mrip(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }

    pred_cL_L_knum(iyear)=sum(L_cL_num(iyear))/1000.0;
    pred_cP_L_knum(iyear)=sum(L_cP_num(iyear))/1000.0;
    pred_cT_L_knum(iyear)=sum(L_cT_num(iyear))/1000.0;
    pred_HB_L_knum(iyear)=sum(L_HB_num(iyear))/1000.0;
    pred_mrip_L_knum(iyear)=sum(L_mrip_num(iyear))/1000.0;
  }


FUNCTION get_landings_wgt

////---Predicted landings------------------------
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    L_cL_klb(iyear)=elem_prod(L_cL_num(iyear),wgt_cL_klb(iyear));     //in 1000 lb
    L_cP_klb(iyear)=elem_prod(L_cP_num(iyear),wgt_cP_klb(iyear));     //in 1000 lb
    L_cT_klb(iyear)=elem_prod(L_cT_num(iyear),wgt_cT_klb(iyear));     //in 1000 lb
```

```
    L_HB_klb(iyear)=elem_prod(L_HB_num(iyear),wgt_HB_klb(iyear));     //in 1000 lb
    L_mrip_klb(iyear)=elem_prod(L_mrip_num(iyear),wgt_mrip_klb(iyear));  //in 1000 lb

    pred_cL_L_klb(iyear)=sum(L_cL_klb(iyear));
    pred_cP_L_klb(iyear)=sum(L_cP_klb(iyear));
    pred_cT_L_klb(iyear)=sum(L_cT_klb(iyear));
    pred_HB_L_klb(iyear)=sum(L_HB_klb(iyear));
    pred_mrip_L_klb(iyear)=sum(L_mrip_klb(iyear));
  }



FUNCTION get_dead_discards //Baranov catch eqn
  //dead discards at age (number fish)

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      D_comm_num(iyear,iage)=N(iyear,iage)*F_comm_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      D_HB_num(iyear,iage)=N(iyear,iage)*F_HB_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      D_mrip_num(iyear,iage)=N(iyear,iage)*F_mrip_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_comm_D_knum(iyear)=sum(D_comm_num(iyear))/1000.0;         //pred annual dead discards in 1000s (for matching data)
    pred_comm_D_klb(iyear)=sum(elem_prod(D_comm_num(iyear),wgt_comm_D_klb(iyear)));  //annual dead discards in 1000 lb (for output only)

    pred_HB_D_knum(iyear)=sum(D_HB_num(iyear))/1000.0;             //pred annual dead discards in 1000s (for matching data)
    pred_HB_D_klb(iyear)=sum(elem_prod(D_HB_num(iyear),wgt_HB_D_klb(iyear)));        //annual dead discards in 1000 lb (for output only)

    pred_mrip_D_knum(iyear)=sum(D_mrip_num(iyear))/1000.0;         //pred annual dead discards in 1000s (for matching data)
    pred_mrip_D_klb(iyear)=sum(elem_prod(D_mrip_num(iyear),wgt_mrip_D_klb(iyear)));  //annual dead discards in 1000 lb (for output only)
  }

FUNCTION get_catchability_fcns
 //Get rate increase if estimated, otherwise fixed above
  if (set_q_rate_phase>0.0)
  {
      for (iyear=styr_cL_cpue; iyear<=endyr_cL_cpue; iyear++)
      {   if (iyear>styr_cL_cpue & iyear <=2003)
          {//q_rate_fcn_cL(iyear)=(1.0+q_rate)*q_rate_fcn_cL(iyear-1); //compound
             q_rate_fcn_cL(iyear)=(1.0+(iyear-styr_cL_cpue)*q_rate)*q_rate_fcn_cL(styr_cL_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_cL(iyear)=q_rate_fcn_cL(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_HBD_cpue; iyear<=endyr_HBD_cpue; iyear++)
      {   if (iyear>styr_HBD_cpue & iyear <=2003)
          {//q_rate_fcn_HBD(iyear)=(1.0+q_rate)*q_rate_fcn_HBD(iyear-1); //compound
             q_rate_fcn_HBD(iyear)=(1.0+(iyear-styr_HBD_cpue)*q_rate)*q_rate_fcn_HBD(styr_HBD_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_HBD(iyear)=q_rate_fcn_HBD(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(B0_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*B0_q_DD))); }
  }


FUNCTION get_indices
//---Predicted CPUEs------------------------
 //Survey 1: Mbft
  for (iyear=styr_Mbft_cpue; iyear<=endyr_Mbft_cpue; iyear++)
  {   //index in number units
      N_Mbft(iyear)=elem_prod(N_mdyr(iyear),sel_Mbft(iyear));
      pred_Mbft_cpue(iyear)=mfexp(log_q_Mbft)*sum(N_Mbft(iyear));
  }

 //Survey 2: Mcvt
  for (iyear=styr_Mcvt_cpue; iyear<=endyr_Mcvt_cpue; iyear++)
  {   //index in number units
      N_Mcvt(iyear)=elem_prod(N_mdyr(iyear),sel_Mcvt(iyear));
      pred_Mcvt_cpue(iyear)=mfexp(log_q_Mcvt)*sum(N_Mcvt(iyear));
  }

 //Commercial handline cpue
  q_cL(styr_cL_cpue)=mfexp(log_q_cL);
  for (iyear=styr_cL_cpue; iyear<=endyr_cL_cpue; iyear++)
  {   //index in weight units. original index in lb and re-scaled. predicted in klb, but difference is absorbed by q
      N_cL(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_cL(iyear)),wgt_cL_klb(iyear));
      pred_cL_cpue(iyear)=q_cL(iyear)*q_rate_fcn_cL(iyear)*q_DD_fcn(iyear)*sum(N_cL(iyear));
      if (iyear<endyr_cL_cpue){q_cL(iyear+1)=q_cL(iyear)*mfexp(q_RW_log_dev_cL(iyear));}
  }

 //Headboat cpue
  q_HB(styr_HB_cpue)=mfexp(log_q_HB);
  for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
  {   //index in weight units. original index in lb and re-scaled. predicted in klb, but difference is absorbed by q
      N_HB(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_HB(iyear)),wgt_HB_klb(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));}
  }
  //HBD cpue
  q_HBD(styr_HBD_cpue)=mfexp(log_q_HBD);
  for (iyear=styr_HBD_cpue; iyear<=endyr_HBD_cpue; iyear++)
  {   //index in number units
      N_HBD(iyear)=elem_prod(N_mdyr(iyear),sel_HB_D(iyear));
      pred_HBD_cpue(iyear)=q_HBD(iyear)*q_rate_fcn_HBD(iyear)*q_DD_fcn(iyear)*sum(N_HBD(iyear));
      if (iyear<endyr_HBD_cpue){q_HBD(iyear+1)=q_HBD(iyear)*mfexp(q_RW_log_dev_HBD(iyear));}
  }

FUNCTION get_length_comps

  //Mbft
  for (iyear=1;iyear<=nyr_Mbft_lenc;iyear++)
      {pred_Mbft_lenc(iyear)=(N_Mbft(yrs_Mbft_lenc(iyear))*lenprob)/sum(N_Mbft(yrs_Mbft_lenc(iyear)));}

  //Commercial lines
  for (iyear=1;iyear<=nyr_cL_lenc;iyear++) //all yrs within periods 2,3
  {  if (yrs_cL_lenc(iyear)<=endyr_period2)
     {pred_cL_lenc(iyear)=(L_cL_num(yrs_cL_lenc(iyear))*lenprob_cL2)
                          /sum(L_cL_num(yrs_cL_lenc(iyear)));
     }
     if (yrs_cL_lenc(iyear)>endyr_period2)
     {pred_cL_lenc(iyear)=(L_cL_num(yrs_cL_lenc(iyear))*lenprob_cL3)
                          /sum(L_cL_num(yrs_cL_lenc(iyear)));
     }
  }

  //Commercial pots: pooled all from period 2
  L_cP_num_pool.initialize();
  for (iyear=1;iyear<=nyr_cP_lenc_pool;iyear++)
  {  L_cP_num_pool_yr(iyear)=nsamp_cP_lenc_pool(iyear)*L_cP_num(yrs_cP_lenc_pool(iyear))
                             /sum(L_cP_num(yrs_cP_lenc_pool(iyear)));
     if (yrs_cP_lenc_pool(iyear)<=endyr_period2) {L_cP_num_pool(1)+=L_cP_num_pool_yr(iyear);}
  }
  for (iyear=1;iyear<=nyr_cP_lenc;iyear++) //all yrs within periods 2
  {  if (yrs_cP_lenc(iyear)<=endyr_period2)
       {pred_cP_lenc(iyear)=(L_cP_num_pool(iyear)*lenprob_cP2)/sum(L_cP_num_pool(iyear)); }
  }



 //Headboat
  for (iyear=1;iyear<=nyr_HB_lenc;iyear++)  //all in periods 1,2,3
  {  if (yrs_HB_lenc(iyear)<=endyr_period1)
     {pred_HB_lenc(iyear)=(L_HB_num(yrs_HB_lenc(iyear))*lenprob_HB1)
                          /sum(L_HB_num(yrs_HB_lenc(iyear)));
     }
     if (yrs_HB_lenc(iyear)>endyr_period1 & yrs_HB_lenc(iyear)<=endyr_period2)
     {pred_HB_lenc(iyear)=(L_HB_num(yrs_HB_lenc(iyear))*lenprob_HB2)
                          /sum(L_HB_num(yrs_HB_lenc(iyear)));
     }
     if (yrs_HB_lenc(iyear)>endyr_period2)
     {pred_HB_lenc(iyear)=(L_HB_num(yrs_HB_lenc(iyear))*lenprob_HB3)
                          /sum(L_HB_num(yrs_HB_lenc(iyear)));
     }
  }
 //HB discards
  for (iyear=1;iyear<=nyr_HB_D_lenc;iyear++) //all yrs within period 3,4
  {  if (yrs_HB_D_lenc(iyear)<=endyr_recr_period3)
     {pred_HB_D_lenc(iyear)=(D_HB_num(yrs_HB_D_lenc(iyear))*lenprob_HB_D3)
                         /sum(D_HB_num(yrs_HB_D_lenc(iyear)));}
    if (yrs_HB_D_lenc(iyear)>endyr_recr_period3)
     {pred_HB_D_lenc(iyear)=(D_HB_num(yrs_HB_D_lenc(iyear))*lenprob_HB_D4)
                         /sum(D_HB_num(yrs_HB_D_lenc(iyear)));}
  }


 //MRIP
  for (iyear=1;iyear<=nyr_mrip_lenc;iyear++)  //all in periods 1,2,3
  {  if (yrs_mrip_lenc(iyear)<=endyr_period1)
     {pred_mrip_lenc(iyear)=(L_mrip_num(yrs_mrip_lenc(iyear))*lenprob_mrip1)
                          /sum(L_mrip_num(yrs_mrip_lenc(iyear)));
     }
     if (yrs_mrip_lenc(iyear)>endyr_period1 & yrs_mrip_lenc(iyear)<=endyr_period2)
     {pred_mrip_lenc(iyear)=(L_mrip_num(yrs_mrip_lenc(iyear))*lenprob_mrip2)
                          /sum(L_mrip_num(yrs_mrip_lenc(iyear)));
     }
     if (yrs_mrip_lenc(iyear)>endyr_period2 & yrs_mrip_lenc(iyear)<=endyr_recr_period3)
     {pred_mrip_lenc(iyear)=(L_mrip_num(yrs_mrip_lenc(iyear))*lenprob_mrip3)
                          /sum(L_mrip_num(yrs_mrip_lenc(iyear)));
     }
     if (yrs_mrip_lenc(iyear)>endyr_recr_period3)
     {pred_mrip_lenc(iyear)=(L_mrip_num(yrs_mrip_lenc(iyear))*lenprob_mrip4)
                          /sum(L_mrip_num(yrs_mrip_lenc(iyear)));
     }
  }


FUNCTION get_age_comps

 //MARMAP bft
  for (iyear=1;iyear<=nyr_Mbft_agec;iyear++)
  {
    ErrorFree_Mbft_agec(iyear)=N_Mbft(yrs_Mbft_agec(iyear))/sum(N_Mbft(yrs_Mbft_agec(iyear)));
    pred_Mbft_agec(iyear)=age_error*ErrorFree_Mbft_agec(iyear);
  }

 //MARMAP cvt
  for (iyear=1;iyear<=nyr_Mcvt_agec;iyear++)
  {
    ErrorFree_Mcvt_agec(iyear)=N_Mcvt(yrs_Mcvt_agec(iyear))/sum(N_Mcvt(yrs_Mcvt_agec(iyear)));
    pred_Mcvt_agec(iyear)=age_error*ErrorFree_Mcvt_agec(iyear);
```

```
  }

 //Commercial lines
  for (iyear=1;iyear<=nyr_cL_agec;iyear++)
  {
    ErrorFree_cL_agec(iyear)=L_cL_num(yrs_cL_agec(iyear))/sum(L_cL_num(yrs_cL_agec(iyear)));
    pred_cL_agec(iyear)=age_error*ErrorFree_cL_agec(iyear);
  }

 //Commercial pots
  for (iyear=1;iyear<=nyr_cP_agec;iyear++)
  {
    ErrorFree_cP_agec(iyear)=L_cP_num(yrs_cP_agec(iyear))/sum(L_cP_num(yrs_cP_agec(iyear)));
    pred_cP_agec(iyear)=age_error*ErrorFree_cP_agec(iyear);
  }

 //Headboat
  for (iyear=1;iyear<=nyr_HB_agec;iyear++)
  {
    ErrorFree_HB_agec(iyear)=L_HB_num(yrs_HB_agec(iyear))/sum(L_HB_num(yrs_HB_agec(iyear)));
    pred_HB_agec(iyear)=age_error*ErrorFree_HB_agec(iyear);
  }

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


////--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cL+
        sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cP+
        sum(log_F_dev_cP((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_HB+
        sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  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_comm_D+
        sum(log_F_dev_comm_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_HB_D+
        sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_mrip_D+
        sum(log_F_dev_mrip_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);

  F_cL_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cL+
        sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cP_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cP+
        sum(log_F_dev_cP((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_HB_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_HB+
        sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  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_comm_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_comm_D+
        sum(log_F_dev_comm_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_HB_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_HB_D+
        sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_mrip_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_mrip_D+
        sum(log_F_dev_mrip_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;

  log_F_dev_end_cL=sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_cP=sum(log_F_dev_cP((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_HB=sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  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_comm_D=sum(log_F_dev_comm_D((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_HB_D=sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_mrip_D=sum(log_F_dev_mrip_D((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

  F_end_L=sel_cL(endyr)*mfexp(log_avg_F_cL+log_F_dev_end_cL)+
          sel_cP(endyr)*mfexp(log_avg_F_cP+log_F_dev_end_cP)+
          sel_HB(endyr)*mfexp(log_avg_F_HB+log_F_dev_end_HB)+
          sel_mrip(endyr)*mfexp(log_avg_F_mrip+log_F_dev_end_mrip);

  F_end_D=sel_comm_D(endyr)*mfexp(log_avg_F_comm_D+log_F_dev_end_comm_D)+
          sel_HB_D(endyr)*mfexp(log_avg_F_HB_D+log_F_dev_end_HB_D)+
          sel_mrip_D(endyr)*mfexp(log_avg_F_mrip_D+log_F_dev_end_mrip_D);

  F_end=F_end_L+F_end_D;
  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));
  sel_wgted_D=elem_prod(sel_wgted_tot, elem_div(F_end_D,F_end));

  wgt_wgted_L_denom=F_cL_prop+F_cP_prop+F_HB_prop+F_mrip_prop;
  wgt_wgted_L_klb=F_cL_prop/wgt_wgted_L_denom*wgt_cL_klb(endyr)+
              F_cP_prop/wgt_wgted_L_denom*wgt_cP_klb(endyr)+
              F_HB_prop/wgt_wgted_L_denom*wgt_HB_klb(endyr)+
              F_mrip_prop/wgt_wgted_L_denom*wgt_mrip_klb(endyr);

  wgt_wgted_D_denom=F_comm_D_prop+F_HB_D_prop+F_mrip_D_prop;
  wgt_wgted_D_klb=F_comm_D_prop/wgt_wgted_D_denom*wgt_comm_D_klb(endyr)+
              F_HB_D_prop/wgt_wgted_D_denom*wgt_HB_D_klb(endyr)+
              F_mrip_D_prop/wgt_wgted_D_denom*wgt_mrip_D_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_D_age_msy=0.0;

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


    //Compute equilibrium values of R (including bias correction), SSB and Yield at each F
    R_eq(ff)=(R0/((5.0*steep-1.0)*spr_msy(ff)))*
                  (BiasCor*4.0*steep*spr_msy(ff)-spr_F0*(1.0-steep));
    if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
    N_age_msy*=R_eq(ff);
    N_age_msy_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)));
      D_age_msy(iage)=N_age_msy(iage)*(F_D_age_msy(iage)/Z_age_msy(iage))*
                      (1.-mfexp(-1.0*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;
    D_eq_klb(ff)=sum(elem_prod(D_age_msy,wgt_wgted_D_klb));
    D_eq_knum(ff)=sum(D_age_msy)/1000.0;
  }  //end ff loop

  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);
        D_msy_knum_out=D_eq_knum(ff);
        D_msy_klb_out=D_eq_klb(ff);
        F_msy_out=F_msy(ff);
        spr_msy_out=spr_msy(ff);
      }
  }

//--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_miscellaneous_stuff

  sigma_rec_dev=sqrt(var_rec_dev); //sample SD of predicted residuals (may not equal rec_sigma)
  //len_cv=elem_div(len_sd,meanlen_TL);

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

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_klb_yr(iyear)= pred_cL_L_klb(iyear)+pred_cP_L_klb(iyear)+pred_cT_L_klb(iyear)+
                               pred_HB_L_klb(iyear)+pred_mrip_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_cL_L_knum(iyear)+pred_cP_L_knum(iyear)+pred_cT_L_knum(iyear)+
                               pred_HB_L_knum(iyear)+pred_mrip_L_knum(iyear);

        D_total_knum_yr(iyear)+=pred_comm_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_comm_D_klb(iyear);

        D_total_knum_yr(iyear)+=pred_HB_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_HB_D_klb(iyear);

        D_total_knum_yr(iyear)+=pred_mrip_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_mrip_D_klb(iyear);

        D_comm_klb(iyear)=elem_prod(D_comm_num(iyear),wgt_comm_D_klb(iyear));     //in 1000 lb
        D_HB_klb(iyear)=elem_prod(D_HB_num(iyear),wgt_HB_D_klb(iyear));          //in 1000 lb
        D_mrip_klb(iyear)=elem_prod(D_mrip_num(iyear),wgt_mrip_D_klb(iyear));    //in 1000 lb

        B(iyear)=elem_prod(N(iyear),wgt_mt);
```

```
        totN(iyear)=sum(N(iyear));
        totB(iyear)=sum(B(iyear));
  }

  L_total_num=(L_cL_num+L_cP_num+L_cT_num+L_HB_num+L_mrip_num); //landings at age in number fish
  L_total_klb=L_cL_klb+L_cP_klb+L_cT_klb+L_HB_klb+L_mrip_klb;   //landings at age in klb whole weight

  D_total_num=(D_comm_num+D_HB_num+D_mrip_num);          //discards at age in number fish
  D_total_klb=D_comm_klb+D_HB_klb+D_mrip_klb;            //discards at age in klb whole weight

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

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

  if(F_msy_out>0)
    {
      FdF_msy=Fapex/F_msy_out;
      FdF_msy_end=FdF_msy(endyr);
      FdF_msy_end_mean=pow((FdF_msy(endyr)*FdF_msy(endyr-1)),(1.0/2.0));
      //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; iyear++)
     {log_rec_dev_output(iyear)=log_rec_dev(iyear);}
   //fill in log Nage deviations for ages they are nonzero (ages2+)
   for(iage=2; iage<=nages; iage++)
   { log_Nage_dev_output(iage)=log_Nage_dev(iage);}


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

  //static per-recruit stuff

  for(iyear=styr; iyear<=endyr; iyear++)
  {
    N_age_spr(1)=1.0;
    for(iage=2; iage<=nages; iage++)
    {
      N_age_spr(iage)=N_age_spr(iage-1)*mfexp(-1.*Z(iyear,iage-1));
    }
    N_age_spr(nages)=N_age_spr(nages)/(1.0-mfexp(-1.*Z(iyear,nages)));
    N_age_spr_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+F_spr(ff)*sel_wgted_D;

    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_Mbft_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
  neff_cL_lenc_allyr_out=missing;
  neff_cP_lenc_allyr_out=missing;
  neff_HB_lenc_allyr_out=missing;
  neff_HB_D_lenc_allyr_out=missing;
  neff_mrip_lenc_allyr_out=missing;
  neff_Mbft_agec_allyr_out=missing;
  neff_Mcvt_agec_allyr_out=missing;
  neff_cL_agec_allyr_out=missing;
  neff_cP_agec_allyr_out=missing;
```

```
  neff_HB_agec_allyr_out=missing;
  neff_mrip_agec_allyr_out=missing;

      for (iyear=1; iyear<=nyr_Mbft_lenc; iyear++)
         {if (nsamp_Mbft_lenc(iyear)>=minSS_lenc)
            {neff_Mbft_lenc_allyr_out(yrs_Mbft_lenc(iyear))=multinom_eff_N(pred_Mbft_lenc(iyear),obs_Mbft_lenc(iyear));}
          else {neff_Mbft_lenc_allyr_out(yrs_Mbft_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cL_lenc; iyear++)
         {if (nsamp_cL_lenc(iyear)>=minSS_lenc)
            {neff_cL_lenc_allyr_out(yrs_cL_lenc(iyear))=multinom_eff_N(pred_cL_lenc(iyear),obs_cL_lenc(iyear));}
          else {neff_cL_lenc_allyr_out(yrs_cL_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cP_lenc; iyear++)
         {if (nsamp_cP_lenc(iyear)>=minSS_lenc)
             {neff_cP_lenc_allyr_out(yrs_cP_lenc(iyear))=multinom_eff_N(pred_cP_lenc(iyear),obs_cP_lenc(iyear));}
          else {neff_cP_lenc_allyr_out(yrs_cP_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
         {if (nsamp_HB_lenc(iyear)>=minSS_lenc)
             {neff_HB_lenc_allyr_out(yrs_HB_lenc(iyear))=multinom_eff_N(pred_HB_lenc(iyear),obs_HB_lenc(iyear));}
          else {neff_HB_lenc_allyr_out(yrs_HB_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_HB_D_lenc; iyear++)
         {if (nsamp_HB_D_lenc(iyear)>=minSS_lenc)
            {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=multinom_eff_N(pred_HB_D_lenc(iyear),obs_HB_D_lenc(iyear));}
           else {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_mrip_lenc; iyear++)
         {if (nsamp_mrip_lenc(iyear)>=minSS_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_Mbft_agec; iyear++)
         {if (nsamp_Mbft_agec(iyear)>=minSS_agec)
            {neff_Mbft_agec_allyr_out(yrs_Mbft_agec(iyear))=multinom_eff_N(pred_Mbft_agec(iyear),obs_Mbft_agec(iyear));}
          else {neff_Mbft_agec_allyr_out(yrs_Mbft_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_Mcvt_agec; iyear++)
         {if (nsamp_Mcvt_agec(iyear)>=minSS_agec)
            {neff_Mcvt_agec_allyr_out(yrs_Mcvt_agec(iyear))=multinom_eff_N(pred_Mcvt_agec(iyear),obs_Mcvt_agec(iyear));}
          else {neff_Mcvt_agec_allyr_out(yrs_Mcvt_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cL_agec; iyear++)
         {if (nsamp_cL_agec(iyear)>=minSS_agec)
            {neff_cL_agec_allyr_out(yrs_cL_agec(iyear))=multinom_eff_N(pred_cL_agec(iyear),obs_cL_agec(iyear));}
          else {neff_cL_agec_allyr_out(yrs_cL_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cP_agec; iyear++)
         {if (nsamp_cP_agec(iyear)>=minSS_agec)
            {neff_cP_agec_allyr_out(yrs_cP_agec(iyear))=multinom_eff_N(pred_cP_agec(iyear),obs_cP_agec(iyear));}
          else {neff_cP_agec_allyr_out(yrs_cP_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_HB_agec; iyear++)
         {if (nsamp_HB_agec(iyear)>=minSS_agec)
            {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=multinom_eff_N(pred_HB_agec(iyear),obs_HB_agec(iyear));}
          else {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_mrip_agec; iyear++)
         {if (nsamp_mrip_agec(iyear)>=minSS_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;}
         }

//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

FUNCTION evaluate_objective_function
  fval=0.0;
  fval_data=0.0;

//---likelihoods--------------------------

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

  f_Mbft_cpue=0.0;
  f_Mbft_cpue=lk_lognormal(pred_Mbft_cpue, obs_Mbft_cpue, Mbft_cpue_cv, w_I_Mbft);
  fval+=f_Mbft_cpue;
  fval_data+=f_Mbft_cpue;

  f_Mcvt_cpue=0.0;
  f_Mcvt_cpue=lk_lognormal(pred_Mcvt_cpue, obs_Mcvt_cpue, Mcvt_cpue_cv, w_I_Mcvt);
  fval+=f_Mcvt_cpue;
  fval_data+=f_Mcvt_cpue;

  f_cL_cpue=0.0;
  f_cL_cpue=lk_lognormal(pred_cL_cpue, obs_cL_cpue, cL_cpue_cv, w_I_cL);
  fval+=f_cL_cpue;
  fval_data+=f_cL_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_HBD_cpue=0.0;
  f_HBD_cpue=lk_lognormal(pred_HBD_cpue, obs_HBD_cpue, HBD_cpue_cv, w_I_HBD);
  fval+=f_HBD_cpue;
  fval_data+=f_HBD_cpue;

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

  //f_cL_L in 1000 lb ww
  f_cL_L=lk_lognormal(pred_cL_L_klb(styr_cL_L,endyr_cL_L), obs_cL_L(styr_cL_L,endyr_cL_L),
                      cL_L_cv(styr_cL_L,endyr_cL_L), w_L);
  fval+=f_cL_L;
  fval_data+=f_cL_L;
  //f_cP_L in 1000 lb ww
  f_cP_L=lk_lognormal(pred_cP_L_klb(styr_cP_L,endyr_cP_L), obs_cP_L(styr_cP_L,endyr_cP_L),
                      cP_L_cv(styr_cP_L,endyr_cP_L), w_L);
  fval+=f_cP_L;
  fval_data+=f_cP_L;
  //f_cT_L in 1000 lb ww
  f_cT_L=lk_lognormal(pred_cT_L_klb(styr_cT_L,endyr_cT_L), obs_cT_L(styr_cT_L,endyr_cT_L),
                      cT_L_cv(styr_cT_L,endyr_cT_L), w_L);
  fval+=f_cT_L;
  fval_data+=f_cT_L;
  //f_HB_L in 1000 lb
  f_HB_L=lk_lognormal(pred_HB_L_klb(styr_HB_L,endyr_HB_L), obs_HB_L(styr_HB_L,endyr_HB_L),
                      HB_L_cv(styr_HB_L,endyr_HB_L), w_L);
  fval+=f_HB_L;
  fval_data+=f_HB_L;

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


//---Discards-------------------------------

  //f_comm_D in 1000 fish
  f_comm_D=lk_lognormal(pred_comm_D_knum(styr_cL_D,endyr_cL_D), obs_comm_D(styr_cL_D,endyr_cL_D),
                      comm_D_cv(styr_cL_D,endyr_cL_D), w_D);
  fval+=f_comm_D;
  fval_data+=f_comm_D;


  //f_HB_D in 1000 fish
  f_HB_D=lk_lognormal(pred_HB_D_knum(styr_HB_D,endyr_HB_D), obs_HB_D(styr_HB_D,endyr_HB_D),
                        HB_D_cv(styr_HB_D,endyr_HB_D), w_D);
  fval+=f_HB_D;
  fval_data+=f_HB_D;

  //f_mrip_D in 1000 fish
  f_mrip_D=lk_lognormal(pred_mrip_D_knum(styr_mrip_D,endyr_mrip_D), obs_mrip_D(styr_mrip_D,endyr_mrip_D),
                      mrip_D_cv(styr_mrip_D,endyr_mrip_D), w_D);
  fval+=f_mrip_D;
  fval_data+=f_mrip_D;


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

  //f_Mbft_lenc
  f_Mbft_lenc=lk_multinomial(nsamp_Mbft_lenc, pred_Mbft_lenc, obs_Mbft_lenc, nyr_Mbft_lenc, minSS_lenc, w_lc_Mbft);
  fval+=f_Mbft_lenc;
  fval_data+=f_Mbft_lenc;

  //f_cL_lenc
  f_cL_lenc=lk_multinomial(nsamp_cL_lenc, pred_cL_lenc, obs_cL_lenc, nyr_cL_lenc, minSS_lenc, w_lc_cL);
  fval+=f_cL_lenc;
  fval_data+=f_cL_lenc;

  //f_cP_lenc
  f_cP_lenc=lk_multinomial(nsamp_cP_lenc, pred_cP_lenc, obs_cP_lenc, nyr_cP_lenc, minSS_lenc, w_lc_cP);
  fval+=f_cP_lenc;
  fval_data+=f_cP_lenc;

  //f_HB_lenc
  f_HB_lenc=lk_multinomial(nsamp_HB_lenc, pred_HB_lenc, obs_HB_lenc, nyr_HB_lenc, minSS_lenc, w_lc_HB);
  fval+=f_HB_lenc;
  fval_data+=f_HB_lenc;

  //f_HB_D_lenc
  f_HB_D_lenc=lk_multinomial(nsamp_HB_D_lenc, pred_HB_D_lenc, obs_HB_D_lenc, nyr_HB_D_lenc, minSS_lenc, w_lc_HB_D);
  fval+=f_HB_D_lenc;
  fval_data+=f_HB_D_lenc;

  //f_mrip_lenc
  f_mrip_lenc=lk_multinomial(nsamp_mrip_lenc, pred_mrip_lenc, obs_mrip_lenc, nyr_mrip_lenc, minSS_lenc, w_lc_mrip);
  fval+=f_mrip_lenc;
  fval_data+=f_mrip_lenc;


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

  //f_Mbft_agec
  f_Mbft_agec=lk_multinomial(nsamp_Mbft_agec, pred_Mbft_agec, obs_Mbft_agec, nyr_Mbft_agec, minSS_agec, w_ac_Mbft);
  fval+=f_Mbft_agec;
  fval_data+=f_Mbft_agec;

  //f_Mcvt_agec
  f_Mcvt_agec=lk_multinomial(nsamp_Mcvt_agec, pred_Mcvt_agec, obs_Mcvt_agec, nyr_Mcvt_agec, minSS_agec, w_ac_Mcvt);
  fval+=f_Mcvt_agec;
```

```
  fval_data+=f_Mcvt_agec;

  //f_cL_agec
  f_cL_agec=lk_multinomial(nsamp_cL_agec, pred_cL_agec, obs_cL_agec, nyr_cL_agec, minSS_agec, w_ac_cL);
  fval+=f_cL_agec;
  fval_data+=f_cL_agec;

  //f_cP_agec
  f_cP_agec=lk_multinomial(nsamp_cP_agec, pred_cP_agec, obs_cP_agec, nyr_cP_agec, minSS_agec, w_ac_cP);
  fval+=f_cP_agec;
  fval_data+=f_cP_agec;

  //f_HB_agec
  f_HB_agec=lk_multinomial(nsamp_HB_agec, pred_HB_agec, obs_HB_agec, nyr_HB_agec, minSS_agec, w_ac_HB);
  fval+=f_HB_agec;
  fval_data+=f_HB_agec;

  //f_mrip_agec
  f_mrip_agec=lk_multinomial(nsamp_mrip_agec, pred_mrip_agec, obs_mrip_agec, nyr_mrip_agec, minSS_agec, w_ac_mrip);
  fval+=f_mrip_agec;
  fval_data+=f_mrip_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_sqd2)/(2.0*rec_sigma_sq));
  for(iyear=(styr_rec_dev+1); iyear<=endyr; iyear++)
  {f_rec_dev+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sqd2)/
              (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)
        {
          f_rec_dev_early=(square(log_rec_dev(styr_rec_dev) + rec_sigma_sq/2.0)/(2.0*rec_sigma_sq)) + rec_logL_add;
          for(iyear=(styr_rec_dev+1); iyear<=endyr_rec_phase1; iyear++)
          {f_rec_dev_early+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sqd2)/
                            (2.0*rec_sigma_sq)) + rec_logL_add;}
        }
  fval+=w_rec_early*f_rec_dev_early;
  }

  f_rec_dev_end=0.0; //possible extra constraint on ending rec deviations
  if (w_rec_end>0.0)
  { if (endyr_rec_phase2<endyr)
        {
          for(iyear=(endyr_rec_phase2+1); iyear<=endyr; iyear++)
          {f_rec_dev_end+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sqd2)/
                            (2.0*rec_sigma_sq)) + rec_logL_add;}
        }
       fval+=w_rec_end*f_rec_dev_end;
   }


//  f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
//  if (styr_rec_dev<endyr_rec_phase1)
//    {
//      f_rec_dev_early=pow(log_rec_dev(styr_rec_dev),2);
//      for(iyear=(styr_rec_dev+1); iyear<=endyr_rec_phase1; iyear++)
//      {f_rec_dev_early+=pow((log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1)),2);}
//    }
//  fval+=w_rec_early*f_rec_dev_early;

//  f_rec_dev_end=0.0; //possible extra constraint on ending rec deviations
//  if (endyr_rec_phase2<endyr)
//    {
//      for(iyear=(endyr_rec_phase2+1); iyear<=endyr; iyear++)
//      {f_rec_dev_end+=pow((log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1)),2);}
//    }
//  fval+=w_rec_end*f_rec_dev_end;

//  f_Ftune=0.0;
//  if (set_Ftune>0.0 && !last_phase()) {f_Ftune=square(Fapex(set_Ftune_yr)-set_Ftune);}
//  fval+=w_Ftune*f_Ftune;

//  //code below contingent on four phases
//  f_fullF_constraint=0.0;
//  if (!last_phase())
//  {for (iyear=styr; iyear<=endyr; iyear++)
//       {if (Fapex(iyear)>3.0){f_fullF_constraint+=mfexp(Fapex(iyear)-3.0);}}
//   if (current_phase()==1) {w_fullF=set_w_fullF;}
//   if (current_phase()==2) {w_fullF=set_w_fullF/10.0;}
//   if (current_phase()==3) {w_fullF=set_w_fullF/100.0;}
//  }

//  fval+=w_fullF*f_fullF_constraint;
//
//  f_fullF_constraint=0.0;
//  for (iyear=styr; iyear<=endyr; iyear++)
//       {if (Fapex(iyear)>3.0){f_fullF_constraint+=mfexp(Fapex(iyear)-3.0);}}
//  fval+=w_fullF*f_fullF_constraint;

//  f_cvlen_diff_constraint=0.0;
//    f_cvlen_diff_constraint=norm2(first_difference(log_len_cv_dev));
//  fval+=w_cvlen_diff*f_cvlen_diff_constraint;
//
//  f_cvlen_dev_constraint=0.0;
//    f_cvlen_dev_constraint=norm2(log_len_cv_dev);
//  fval+=w_cvlen_dev*f_cvlen_dev_constraint;
```

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

  //Random walk components of fishery dependent indices: these components equal zero if RW turned off
  f_cL_RW_cpue=0.0;
  for (iyear=styr_cL_cpue; iyear<endyr_cL_cpue; iyear++)
      {f_cL_RW_cpue+=square(q_RW_log_dev_cL(iyear))/(2.0*set_q_RW_cL_var);}
  fval+=f_cL_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_HBD_RW_cpue=0.0;
  for (iyear=styr_HBD_cpue; iyear<endyr_HBD_cpue; iyear++)
      {f_HBD_RW_cpue+=square(q_RW_log_dev_HBD(iyear))/(2.0*set_q_RW_HBD_var);}
  fval+=f_HBD_RW_cpue;


//---Priors-----------------------------------------------------
//neg_log_prior arguments: estimate, prior, variance, pdf type
//Variance input as a negative value is considered to be CV in arithmetic space (CV=-1 implies loose prior)
//pdf type(1=none, 2=lognormal, 3=normal, 4=beta)

  f_priors=0.0;
  f_priors+=neg_log_prior(steep, set_steep, square(set_steep_se), 4);
  f_priors+=neg_log_prior(rec_sigma,set_rec_sigma,square(set_rec_sigma_se),3);
  f_priors+=neg_log_prior(R_autocorr,set_R_autocorr, 1.0, 1);
  //f_priors+=neg_log_prior(q_DD_beta, set_q_DD_beta, square(set_q_DD_beta_se), 3);
  //f_priors+=neg_log_prior(q_rate, set_q_rate, dzero+square(set_q_rate), 3);
  f_priors+=neg_log_prior(F_init_ratio, set_F_init_ratio, -1.0 , 3);
  //f_priors+=neg_log_prior(M_constant, set_M_constant, square(set_M_constant_se), 3);

  //f_priors+=neg_log_prior(Linf,set_Linf,square(set_Linf_se),3);
  //f_priors+=neg_log_prior(K,set_K,square(set_K_se),3);
  //f_priors+=neg_log_prior(t0,set_K,square(set_t0_se),3);
  f_priors+=neg_log_prior(len_cv_val,set_len_cv,square(set_len_cv_se),3);

  f_priors+=neg_log_prior(selpar_L50_Mbft, set_selpar_L50_Mbft, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_Mbft, set_selpar_slope_Mbft, -1.0, 3);

  f_priors+=neg_log_prior(selpar_L50_Mcvt, set_selpar_L50_Mcvt, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_Mcvt, set_selpar_slope_Mcvt, -1.0, 3);

  f_priors+=neg_log_prior(selpar_L50_cL2, set_selpar_L50_cL2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cL2, set_selpar_slope_cL2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_L50_cL3, set_selpar_L50_cL3, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cL3, set_selpar_slope_cL3, -1.0, 3);

  f_priors+=neg_log_prior(selpar_L50_cP2, set_selpar_L50_cP2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cP2, set_selpar_slope_cP2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_L50_cP3, set_selpar_L50_cP3, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cP3, set_selpar_slope_cP3, -1.0, 3);

  f_priors+=neg_log_prior(selpar_L50_HB1, set_selpar_L50_HB1, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_HB1, set_selpar_slope_HB1, -1.0, 3);
  f_priors+=neg_log_prior(selpar_L50_HB2, set_selpar_L50_HB2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_HB2, set_selpar_slope_HB2, -1.0, 3);
  f_priors+=neg_log_prior(selpar_L50_HB3, set_selpar_L50_HB3, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_HB3, set_selpar_slope_HB3, -1.0, 3);
  f_priors+=neg_log_prior(selpar_L50_HB4, set_selpar_L50_HB4, -1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_HB4, set_selpar_slope_HB4, -1.0, 3);

  f_priors+=neg_log_prior(selpar_Age0_HB_D_logit, set_selpar_Age0_HB_D_logit, -1.0, 3);
  f_priors+=neg_log_prior(selpar_Age1_HB_D_logit, set_selpar_Age1_HB_D_logit, -1.0, 3);
  f_priors+=neg_log_prior(selpar_Age2_HB_D_logit, set_selpar_Age2_HB_D_logit, -1.0, 3);

  fval+=f_priors;

  //cout << "fval = " << fval << "  fval_data = " << fval_data << endl;


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

//--------------------------------------------------------------------------------
//Logistic function: 4 parameters
FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase, L502=age at 50% decrease additive to L501, slope2=slope of decrease
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=elem_prod( (1./(1.+mfexp(-1.*slope1*(ages-L501)))),(1.-(1./(1.+mfexp(-1.*slope2*(ages-(L501+L502)))))) );
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

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

```
   if (double(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope1*(ages(iage)-L501)));}
   if (double(iage)>joint){Sel_Tmp(iage)=1.0-(1.0-satval)/(1.+mfexp(-1.*slope2*(ages(iage)-L502)));}
  }
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

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

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

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

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

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

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

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

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

//--------------------------------------------------------------------------------
//Likelihood contribution: lognormal
FUNCTION dvariable lk_lognormal(const dvar_vector& pred, const dvar_vector& obs, const dvar_vector& cv, const dvariable& wgt_dat)
  //pred=vector of predicted vals, obs=vector of observed vals, cv=vector of CVs in arithmetic space, wgt_dat=constant scaling of CVs
  //dzero is small value to avoid log(0) during search
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  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+dzero),(obs+dzero)))),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;
  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)+dzero),
               log(elem_div((pred_comp(ii)+dzero), (obs_comp(ii)+dzero)))));
    }
  }
  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;
```

```
   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=huge_number;
         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=huge_number;
         break;
       default: // no such prior pdf currently available
         cout << "The prior must be either 1(lognormal), 2(normal), or 3(beta)." << endl;
         cout << "Presently it is " << pdf << endl;
         exit(0);
   }
   return LkvalTmp;


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

  if (last_phase())
  {

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

      cout <<endl;
      cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;
      cout << "BC Fmsy=" << F_msy_out<< "   BC SSBmsy=" << SSB_msy_out <<endl;
      cout <<"F status="<<FdF_msy_end<<endl;
      cout <<"Pop status="<<SdSSB_msy_end<<endl;
      cout << "h="<<steep<<"   R0="<<R0<<endl;
      cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;

      report << "TotalLikelihood " << fval << endl;
      report << "N" << endl;
      report << N<<endl;
      report << "SSB" << endl;
      report << SSB << endl;

     #include "bsb_make_Robject5.cxx"   // write the S-compatible report

  } //endl last phase loop
```