The Beaufort Assessment Model (BAM) with application to red snapper: 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 a statistical catch-age model (Quinn and Deriso 1999), implemented with the AD Model Builder software (ADMB Project 2009). In essence, a statistical catch-age model simulates a population forward in time while including fishing processes. 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 used by Fournier and Archibald (1982), Deriso et al. (1985) in their CAGEAN model, and Methot (1989) in his Stock Synthesis model. The catch-age model of this assessment is similar in structure to the CAGEAN and stock-synthesis models. Previous versions of this assessment model have been used in SEDAR assessments of reef fishes in the U.S. South Atlantic, such as red porgy, black sea bass, tilefish, snowy grouper, gag grouper, greater amberjack, vermilion snapper, Spanish mackerel, and red grouper, as well as red snapper (SEDAR 15). The present version of this code, customized for SEDAR 24 red snapper, is described below.

2 Model configuration and equations

Model equations are detailed in Table 2.1, and AD Model Builder code for implementation is supplied in Appendix A. An input data file for red snapper is included as Appendix B. A general description of the assessment model follows:

Stock dynamics In the assessment model, new biomass was acquired through growth and recruitment, while abundance of existing cohorts experienced exponential decay from fishing and natural mortality. The population was assumed closed to immigration and emigration. The model included age classes $1 - 20^{+}$, where the oldest age class 20^{+} allowed for the accumulation of fish (i.e., plus group).

Initialization period For the SEDAR24 red snapper assessment, initial (1955) numbers at age assumed the stable age structure computed from expected recruitment and the initial, age-specific total mortality rate. That initial mortality was the sum of natural mortality and fishing mortality, where fishing mortality was the product of an initial fishing rate (F_{init}) and catch-weighted average selectivity. The initial fishing rate was chosen using an iterative approach. First, the assessment model was run using the nearly complete catch history (starting from the year 1901) provided by the DW, to indicate a plausible level of biomass depletion in 1955 ($B_{1955}/B_0 \approx 0.8$). Then, F_{init} was adjusted to approximate that level; the value used in the base model run was $F_{init} = 0.02$.

The initial recruitment in 1955 was assumed to be the expected value from the spawner-recruit curve. For the remainder of the initialization period (1955–1975), recruitment was permitted to deviate from the spawner-recruit curve. However, without CPUE or age/length composition data prior to 1976, there is little information to estimate these historic recruitment deviations with accuracy. Thus, the estimates of historic recruitment should not be considered reliable. Instead, the deviations are permitted to allow the model maximum flexibility to match CPUE and age/length composition data near the onset of the assessment period (1976–2009), as well as to minimize influence of the historic (initialization) period on the estimated spawner-recruit curve and thus management benchmarks. For this latter reason, recruitment deviations are estimated in two stanzas, 1956–1974 and 1975–2009. The log recruitment deviations in the early stanza are not constrained to sum to zero (although values are penalized for deviating from zero to provide some response in the likelihood surface). Only recruitment deviations from the second stanza are used in the spawner-recruit lognormal likelihood.

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 through the oldest observed age (54 years) as would occur with constant $M = 0.08$ from the Data Workshop (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. For fitting length composition data, the distribution of size at age was assumed normal with standard deviation estimated by the assessment model. For fishery 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 landings would include only fish of legal size, and length compositions of discards would include only fish below the size limit. Mean length at age of landings and discards were computed from these truncated distributions, and thus average weight at age of landings and discards may differ from that of the population at large.

Maturity Maturity was modeled with the logistic function; parameters for this model were provided by the DW and treated as input to the assessment model.

Spawning biomass Spawning biomass was modeled as the gonad weight of mature female individuals in the population at the time of spawning, where sex ratio at age (50:50) was provided by the DW. For red snapper, peak spawning was considered to occur in late July and the beginning of August.

Recruitment Estimated recruitment of age-1 fish was predicted from spawning biomass using the Beverton–Holt spawner-recruit model. Steepness, h, is a key parameter of this model, and unfortunately it is often difficult to estimate reliably (Conn et al. 2010). In this assessment, many initial attempts to estimate steepness resulted in a value near it's upper bound of 1.0, indicating that the data were insufficient for estimation. Thus, steepness was fixed at $h = 0.85$, the mode of a beta distribution estimated through meta-analysis (SEDAR 2010).

Annual variation from expected recruitment varied with lognormal deviations. The spawner-recruit curve was estimated using the lognormal residuals only from years when composition data could provide information on year-class strength (1975–2009) (further description in above section describing the initialization period).

Landings Time series of landings from four fisheries were modeled: commercial lines, commercial diving, for-hire (headboat, charter boat), and recreational private boats. Landings were modeled with the Baranov catch equation (Baranov 1918) and were fitted in either weight or numbers, depending on how the data were collected (1000 lb whole weight for commercial fleets and 1000 fish for recreational fleets).

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 (described below) and release mortality rates. Discards were assumed to have a mortality probability of 0.48 for commercial lines, 0.41 for the for-hire sector, and 0.39 for private boats, as estimated by the DW.

Fishing For each time series of landings and discard mortalities, a separate full fishing mortality rate (F) was estimated for each year. Age-specific rates were then computed as the product of full F and selectivity at age.

Selectivities Selectivity curves applied to landings 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. Flat-topped selectivities were modeled as a two-parameter logistic function. Dome-shaped selectivities were modeled by combining two logistic functions: a two-parameter logistic function to describe the ascending limb of the curve and a three-parameter logistic function to describe the descending limb. The two functions were joined at the age of full selection, which was fixed for each model run. To model landings, the AW Panel recommended flat-topped selectivity for commercial lines and dome-shaped selectivity for commercial dive, for-hire, and private recreational fleets.

Selectivity of each fleet was fixed within each block of size-limit regulations, but was permitted to vary among blocks where possible or reasonable. Fisheries experienced three blocks of size-limit regulations (no limit prior to 1983, 12-inch limit during 1983–1991, and 20-inch limit during 1992–2009). 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 the private recreational fleet had little age or length composition data, this fleet assumed no change in selectivity with implementation of the 12-inch size limit, but did allow a change with the 20-inch limit. Furthermore, the descending limb of this selectivity mirrored that of the for-hire fleet. With no composition data for commercial dive prior to the last regulatory block, commercial dive selectivity was assumed constant over time. Commercial lines selectivities in the first and second regulatory blocks were set equal, consistent with the DW recommendation that the 12-inch size limit had little effect on commercial line fishing. Selectivities of fishery dependent indices were the same as those of the relevant fleet.

Selectivities of discards were partially estimated, assuming that discards consisted primarily of undersized fish, as implied by observed length compositions of discards. The general approach taken for for-hire discard selectivity was that the value for age 1 was estimated, age 2 was assumed to have full selection, and selectivity for each age 3+ 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, selectivity would change with modification in the size limit. A similar approach was taken for commercial line discard selectivity, but distinct values for age 1 and age 2 were estimated, age 3 was assumed to have full selection, and ages 4+ were set to probabilities of being below the size limit. For private recreational discards, no age or length composition data were available, and thus selectivity of those discards mirrored that of the for-hire fleet.

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 that situation.

Indices of abundance The model was fit to three fishery dependent indices of abundance (commercial lines 1993–2009; headboat 1976–2009; headboat discards 2005–2009). Predicted indices were conditional on selectivity of the survey/gear and were computed from numbers at age at the midpoint of the year or, in the case of commercial lines, weight at age.

Catchability Several options for catchability were available for the red snapper assessment following recommendations of a 2009 SEDAR procedural workshop on catchability (SEDAR Procedural Guidance 2009). In particular, capabilities for including density dependence, linear trends, and random walks were available, as well as time-invariant catchability. Parameters for these models could be estimated or fixed based on a priori considerations. For red snapper, catchability was assumed to increase linearly 2% per year until 2003, after which catchability was constant as recommended by fishermen during SEDAR 19 (SEDAR 2009a).

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. Computed benchmarks included MSY, fishing mortality rate at MSY (F_{MSY}), and spawning biomass (total mature female gonad biomass) at MSY ($\mathrm{SSB}_{\mathrm{MSY}}$). These benchmarks are conditional on the estimated selectivity functions. 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 (geometric) over the last three years of the assessment.

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 fit using lognormal likelihoods. Length and age composition data were fit 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 desired 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 red snapper, CVs of landings and discards (in arithmetic space) were assumed equal to 0.05, to achieve a close fit to these time series yet allowing 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 iteratively until standard deviations of normalized residuals were near 1.0 (SEDAR24-RW03).

In addition to likelihoods, several penalties and prior distributions were included in the compound objective function. In some cases, as with selectivity slope parameters, priors were applied. Variability around the spawner-recruit curve was assumed lognormal. Priors and penalties 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 with little noise (SEDAR 2007). Further, the general model structure has been through multiple SEDAR reviews. As an additional measure of quality control, red snapper 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 red snapper stock dynamics.

References

ADMB Project, 2009. 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.

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.

Deriso, R. B., T. J. I. 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.

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.

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

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. I., and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York.

SEDAR, 2007. SEDAR 15 Stock Assessment Report: South Atlantic Red Snapper.

SEDAR, 2009a. SEDAR 19 Data Workshop Report.

SEDAR, 2010. SEDAR-24-AW-06: Spawner-recruit relationships of demersal marine fishes: Prior distribution of steepness for possible use in SEDAR stock assessments.

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

Table 2.1. General definitions, input data, population model, and negative log-likelihood components of the statistical catch-age model applied to red snapper. 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	$y \in \{1955 \ldots 2009\}$
Index of ages	a	$a \in \{1 \ldots A\}$, where $A = 20^+$
Index of size-limit periods	r	$r \in \{1 \ldots 3\}$ where $1 = 1955 - 1982$ (no size limit), $2 = 1983 - 1991$ (12-inch TL limit), and $3 = 1992 - 2009$ (20-inch limit)
Index of length bins	l	$l \in \{1 \ldots 38\}$
Length bins	l'	$l' \in \{190, 220, \ldots, 1300\text{mm}\}$, with midpoint of 30mm bin used to match length compositions. Largest 10 length bins treated as a plus group, but retained for weight calculations.
Index of fisheries	f	$f \in \{1 \ldots 4\}$ where 1 = commercial lines, 2 = commercial diving, 3 = recreational for-hire (headboat and charter boat), 4 = recreational private boats
Index of discards	d	$d \in \{1 \ldots 3\}$ where 1=commercial lines, 2=recreational for-hire, 3=recreational private boats
Index of CPUE	u	$u \in \{1 \ldots 3\}$ where 1 = commercial lines, 2 = headboat, 3 = headboat discards
Input Data		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages
Proportion mature at age	m_a	Logistic increase with age; assumed constant across years
Spawning date	t_{spawn}	Fraction denoting the proportional time of year when spawning occurs. Set to 0.583 for red snapper by assuming peak spawning occurs in the end of July and beginning of August.
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. Commercial L in whole weight, and rec L in numbers of fish.
CVs of landings	$c^L_{f,y}$	Assumed 0.05 in arithmetic space
Observed abundance indices	$U_{u,y}$	$u = 1$, commercial lines (weight), $y \in \{1993 \ldots 2009\}$ $u = 2$, headboat (numbers), $y \in \{1976 \ldots 2009\}$ $u = 3$, headboat discards (numbers), $y \in \{2005 \ldots 2009\}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
CVs of abundance indices	$c^U_{u,y}$	$u = \{1 \dots 3\}$ as above. Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean
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.
Observed total discards	$D'_{d,y}$	Discards (1000 fish) in year y from fishery d.
Discard mortality rate	δ_d	Proportion discards by fishery d that die. The DW recommended $\delta_d = 0.48$ for d=1, $\delta_d = 0.41$ for d=2, and $\delta_d = 0.39$ for d=3.
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
SD of l_a	$\widehat{c}^{\lambda}_a$	Estimated standard deviation of growth, assumed constant across ages.
Age–length conversion of population	$\psi^u_{a,l}$	$\psi^u_{a,l} = \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a)^2\right)}$, 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	$\psi^L_{f,a,l,y}$	$\psi^L_{f,a,l,y} = \begin{cases} \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a)^2\right)} & : l_a \geq l_{\text{limit}} \\ 0 & : \text{otherwise} \end{cases}$ where l_{limit} is the size limit for fishery f in year y (and would be treated as 0 prior to regulations). Annual matrices $\psi^L_{f,\cdot,\cdot,y}$ are rescaled to sum to one within ages, with the largest 10 length bins fit as a plus group.
Age–length conversion of discards	$\psi^D_{d,a,l,y}$	$\psi^D_{d,a,l,y} = \begin{cases} \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a)^2\right)} & : l_a < l_{\text{limit}} \\ 0 & : \text{otherwise} \end{cases}$ 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
Gonad weight at age of individuals	g_a	Computed from weight at age by $g_a = \varpi_1 w_a^{\varpi_2}$ where ϖ_1 and ϖ_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 $w^{L,D}_{(f,d),a,y} = \theta_1(\xi^{L,D}_{(f,d),a,y})^{\theta_2}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Fishery and index selectivities	$s_{(f,u),a,r}$	$s_{(f,u),a,r} = \begin{cases} \frac{1}{1+\exp\left[-\widehat{\eta}_{1,(f,u),r}\left(a-\widehat{\alpha}_{1,(f,u),r}\right)\right]} & :f=1; u=1 \\ \left(\frac{1}{1+\exp\left[-\widehat{\eta}_{2,(f,u),r}\left(a-\widehat{\alpha}_{2,(f,u),r}\right)\right]}\right) & :f=2,3,4; u=2,3; a<a_{full} \\ 1.0 & :f=2,3,4; u=2,3; a=a_{full} \\ \left(1-\frac{1-\widehat{\vartheta}_{(f,u),r}}{1+\exp\left[-\widehat{\eta}_{3,(f,u),r}\left(a-\widehat{\alpha}_{3,(f,u),r}\right)\right]}\right) & :f=2,3,4; u=2,3; a>a_{full} \end{cases}$ where $\widehat{\eta}_{1,(f,u),r}$, $\widehat{\eta}_{2,(f,u),r}$, $\widehat{\eta}_{3,(f,u),r}$, $\widehat{\alpha}_{1,(f,u),r}$, $\widehat{\alpha}_{2,(f,u),r}$, $\widehat{\alpha}_{3,(f,u),r}$, and $\widehat{\vartheta}_{(f,u),r}$ are estimated parameters and a_{full} is an age of full selection. 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 private boat fleet was assumed to have the same descending limb of the selectivity function as the for-hire fleet for each regulatory period. Commercial lines selectivity was set equal for the first and second regulatory periods, and the commercial diving selectivity was time-invariant.
Discard selectivity	$s'_{d,a,r}$	$s'_{1,1,r}$ and $s'_{1,2,r}$ estimated; $s'_{1,3,r}$ set to 1.0; $s'_{1,4+,r}$ set equal to the age-specific probability of total length below the size limit in period r. $s'_{2,1,r}$ estimated; $s'_{2,2,r}$ set to 1.0; $s'_{2,3+,r}$ set equal to the age-specific probability of total length below the size limit in period r. The discard selectivity for d=3 was assumed the same as the discard selectivity of d=2.
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	$F_y = \max(F_{a,y})$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Abundance at age	$N_{a,y}$	$N_{1,1955} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}\phi_{init}-0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)\phi_{init}}$ $\widehat{N}_{2+,1955}$ equilibrium conditions expected given assumptions about initial fishing mortality (described below) $N_{1,y+1} = \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_{y+1})\text{for} \quad y \geq 1955$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (1 \ldots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ 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 $\varsigma = \exp(\widehat{\sigma}^2/2)$, where $\widehat{\sigma}^2$ is the variance of recruitment deviations. In the SEDAR24 baserun, $h = 0.85$ and $\sigma = 0.6$ were fixed parameters. 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 July and beginning of August to correspond with peak spawning $N''_{a,y} = \exp(-t_{\text{spawn}}Z_{a,y})N_{a,y}$
Unfished abundance at age per recruit at time of spawning	NPR_a	$NPR_1 = 1 \times \exp(-t_{\text{spawn}}M_1)$ $NPR_{a+1} = NPR_a\exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (1 \ldots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_At_{\text{spawn}})]}{1-\exp(-M_A)}$
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	$\phi_0 = \sum_{a=1}^{A} NPR_a\rho_a m_a g_a$ In units of mature female gonad weight.
Initial spawning biomass per recruit	ϕ_{init}	$\phi_{init} = \sum_{a=1}^{A} NPR_a^{init}\rho_a m_a g_a$ In units of mature female gonad weight.
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y}\rho_a m_a g_a$ Also referred to as spawning biomass in units of total mature female gonad biomass.
Initialization mortality at age	Z_a^{init}	$Z_a^{init} = M_a + s_a^{init}\widehat{F}^{init}$ where $\widehat{F}^{init}$ is an estimated initialization F, and s_a^{init} is the initialization selectivity, assumed to be the catch-weighted (1953–1955) selectivities of fleets. In the SEDAR24 baserun, $F^{init} = 0.02$ was fixed.
Initial equilibrium abundance at age	N_a^{equil}	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	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y}L'_{f,a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
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	$D''_{d,a,y}$	$D''_{d,a,y} = w^D_{d,a,y} D'_{d,a,y}$
Index catchability	$q_{u,y}$	$q_{u,1976} = \widehat{q}^0_u f(\text{density})$ $q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ for $y \geq 1976$ 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 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 1976, $f_y(\text{trend})$ follows, $f_y(\text{trend}) = \begin{cases} 1.0 & :y = 1976 \\ f_{y-1}(\text{trend}) * (\text{y} - 1976)\beta_{\text{q}} & :1976 < y \leq 2003 \\ f_{2003}(\text{trend}) & :2003 < y \end{cases}.$ Finally, $f_y(\text{random}) = \exp(\epsilon_{u,y})$ are lognormal catchability deviations which allow for a random walk in catchability when penalties are placed on the $\epsilon_{u,y}$ (see "Objective Function"). In practice, the catchability function $f_y(\text{trend})$ was used as described for the SEDAR 24 red snapper assessment. Density dependence and random walks were not applied in the baserun.
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \begin{cases} \sum_a L''_{f,a,y} & :f = 1,2 \\ \sum_a L'_{f,a,y} & :f = 3,4 \end{cases}$
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	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \begin{cases} \widehat{q}_{u,y} \sum_a w^L_{u,a,y} N'_{a,y} s_{u,a,r} & : \quad u = 1 \\ \widehat{q}_{u,y} \sum_a N'_{a,y} s_{u,a,r} & : \quad u = 2,3 \end{cases}$ 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 30 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	$\Lambda_3 = \sum_{f}\sum_{y}\frac{\left[\log\left((L_{f,y}+x)\big/(\breve{L}_{f,y}+x)\right)\right]^2}{2(\sigma^{L}_{f,y})^2}$ 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	$\Lambda_4 = \sum_{d}\sum_{y}\frac{\left[\log\left((\delta_d D_{d,y}+x)\big/(\breve{D}_{d,y}+x)\right)\right]^2}{2(\sigma^{D}_{d,y})^2}$ 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	$\Lambda_5 = \sum_{u}\sum_{y}\frac{\left[\log\left((U_{u,y}+x)\big/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma^{U}_{u,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma^{U}_{u,y} = \sqrt{\log(1 + (c^{U}_{u,y}/\omega^{U}_{u})^2)}$, with ω^{U}_{u} a preset weight (selected by iterative re-weighting).
Lognormal recruitment deviations	Λ_6	$\Lambda_6 = \omega_6\left[R^2_{1975} + \sum_{y>1975}\frac{[(R_y - \widehat{\varrho}R_{y-1}) + (\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R}\right]$ where R_y are recruitment deviations in log space, $\omega_6 = 1$ is a preset weight, $\widehat{\varrho}$ is the estimated first-order autocorrelation, and $\widehat{\sigma}^2_R$ is the recruitment variance ($\varrho = 0$ and $\sigma = 0.6$ were fixed in the SEDAR24 baserun).
Additional constraint on initial recruitment deviation	Λ_7	$\Lambda_7 = \sum_{y=1956}^{1974} R^2_y + \omega_7\left(R^2_{1975}\right)$ where ω_7 is a preset weight, with ω_7=0.0 for the SEDAR 24 red snapper baserun.
Additional constraint on final recruitment deviations	Λ_8	$\Lambda_8 = \omega_8\left(\sum_{y\geq 2007} R_y - \widehat{\varrho}R_{y-1})^2\right)$ where ω_8 is a preset weight, with ω_8=0.0 for the SEDAR 24 red snapper baserun.
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 24 red snapper baserun, thus ω_9=0.0.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Penalty on initial age structure	Λ_{10}	$\Lambda_{10} = \sum_{a=2}^{A} (\omega_{10}(\widehat{N}_{a,init} - N_a^{equil})^2$ where ω_{10} is a preset weight, with ω_{10}=0.0 for the SEDAR 24 baserun, where initial population assumed the equilibrium age structure. Used in sensitivity runs that initiated later than 1955.
Prior distributions and penalties	Λ_{11}	Several prior distributions were imposed on parameters to keep them in reasonable parameter spaces: $\Lambda_{11} = \frac{(\widehat{h}-E(h))^2}{\sigma_h^2} + \frac{(\widehat{c_a^\lambda}-E(c_a^\lambda))^2}{\sigma_c^2} + (\widehat{\rho_R} - E(\rho_R))^2 + \frac{(\widehat{\sigma_R}-E(\sigma_R))^2}{\sigma_{\sigma_R}^2} + \frac{(\widehat{F}^{init}-E(F^{init})^2}{\sigma_{F^{init}}^2} +$ $\sum_{f,u}[\frac{(\widehat{\eta}_{1,(f,u),r}-E(\eta_{1,(f,u),r})}{\sigma_{\eta_{1,(f,u),r}}}]^2 + \sum_{f,u}[\frac{(\widehat{\eta}_{2,(f,u),r}-E(\eta_{2,(f,u),r})}{\sigma_{\eta_{2,(f,u),r}}}]^2 + \sum_{f,u}[\frac{(\widehat{\eta}_{3,(f,u),r}-E(\eta_{3,(f,u),r})}{\sigma_{\eta_{3,(f,u),r}}}]^2 +$ $\sum_{f,u}[\frac{(\widehat{\alpha}_{1,(f,u),r}-E(\alpha_{1,(f,u),r})}{\sigma_{\alpha_{1,(f,u),r}}}]^2 + \sum_{f,u}[\frac{(\widehat{\alpha}_{2,(f,u),r}-E(\alpha_{2,(f,u),r})}{\sigma_{\alpha_{2,(f,u),r}}}]^2 +$ $\sum_{f,u}[\frac{(\widehat{\alpha}_{3,(f,u),r}-E(\alpha_{3,(f,u),r})}{\sigma_{\alpha_{3,(f,u),r}}}]^2 + \sum_{f,u}[\frac{(\widehat{\vartheta}_{(f,u),r}-E(\vartheta_{(f,u),r})}{\sigma_{\vartheta_{(f,u),r}}}]^2 +$ $[\text{logit}(\widehat{s'}_{1,1,r}) - \text{logit}(E(s'_{1,1,r}))]^2 + [\text{logit}(\widehat{s'}_{1,2,r}) - \text{logit}(E(s'_{1,2,r}))]^2$ $+ [\text{logit}(\widehat{s'}_{2,1,r}) - \text{logit}(E(s'_{2,1,r}))]^2$ where expected values $E(\theta)$ are supplied as input. Other than σ_h, σ_c, and σ_R, standard deviations assumed CV=1 (i.e., diffuse priors), such that σ (denominator) equaled the expected value. For parameters that were fixed, their contribution to the likelihood would be zero. Not all of the above parameters were estimated in all model runs, e.g., several parameters were fixed in the base run and not all selectivity parameters were used for all fleets. In such cases, fixed parameters did not contribute to the likelihood.
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

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR24 Assessment: Red Snapper, September 2010
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

DATA_SECTION

!!cout << "Starting Red Snapper 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;

//3 possible periods of size regs: styr-82 no restrictions, 1983-91 12-inch TL, 1992-08 20-in TL
init_int endyr_period1;
init_int endyr_period2;

init_number limit_12in;  //12 inch limit in mm
init_number limit_20in;  //20 inch limit in mm
init_number limit_disc;  //size limit applied to discards (may differ from fed size limit)

//Total number of ages
init_int nages;

// Vector of ages for age bins
init_ivector 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;

//#############################################################################
//###HBD index (headboat discards from at sea observer program#################
//CPUE
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

//###################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, multiplied by discard mortality for fitting
init_vector cL_D_cv(styr_cL_D,endyr_cL_D);         //vector of CV of discards by year

// Length Compositions (3 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 neff_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 neff_cL_agec(1,nyr_cL_agec);
init_matrix obs_cL_agec(1,nyr_cL_agec,1,nages);

//Length Compositions (3 cm bins) of commercial line discards
init_int nyr_cL_D_lenc;
init_ivector yrs_cL_D_lenc(1,nyr_cL_D_lenc); //represents 2007-2009
init_vector nsamp_cL_D_lenc(1,nyr_cL_D_lenc);
init_vector neff_cL_D_lenc(1,nyr_cL_D_lenc);
init_matrix obs_cL_D_lenc(1,nyr_cL_D_lenc,1,nlenbins);


//#############################################################################
//##Commercial diving fishery
// Landings (1000 lb whole weight)
init_int styr_cD_L;
init_int endyr_cD_L;
init_vector obs_cD_L(styr_cD_L,endyr_cD_L);
init_vector cD_L_cv(styr_cD_L,endyr_cD_L);    //vector of CV of landings by year
// Length Compositions (3 cm bins, data from diving)
init_int nyr_cD_lenc;
init_ivector yrs_cD_lenc(1,nyr_cD_lenc);
init_vector nsamp_cD_lenc(1,nyr_cD_lenc);
init_vector neff_cD_lenc(1,nyr_cD_lenc);
init_matrix obs_cD_lenc(1,nyr_cD_lenc,1,nlenbins);
// Age Compositions (data from diving)
init_int nyr_cD_agec;
init_ivector yrs_cD_agec(1,nyr_cD_agec);
init_vector nsamp_cD_agec(1,nyr_cD_agec);
init_vector neff_cD_agec(1,nyr_cD_agec);
init_matrix obs_cD_agec(1,nyr_cD_agec,1,nages);

//#############################################################################
//################################Headboat+Charterboat (for-hire) fishery ########################################
//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
// Landings (1000 fish)
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 (3 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 neff_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 neff_HB_agec(1,nyr_HB_agec);
init_matrix obs_HB_agec(1,nyr_HB_agec,1,nages);
//Length Compositions (3 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 neff_HB_D_lenc(1,nyr_HB_D_lenc);
init_matrix obs_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);

//
//#############################################################################
//############################PVT landings #################################
// Landings (1000 fish)
init_int styr_PVT_L;
init_int endyr_PVT_L;
init_vector obs_PVT_L(styr_PVT_L,endyr_PVT_L);
init_vector PVT_L_cv(styr_PVT_L,endyr_PVT_L);
// Discards (1000s)
init_int styr_PVT_D;
init_int endyr_PVT_D;
init_vector obs_PVT_released(styr_PVT_D,endyr_PVT_D); //vector of observed releases by year, multiplied by discard mortality for fitting
init_vector PVT_D_cv(styr_PVT_D,endyr_PVT_D);         //vector of CV of discards by year
// Length Compositions (3 cm bins)
init_int nyr_PVT_lenc;
init_ivector yrs_PVT_lenc(1,nyr_PVT_lenc);
init_vector nsamp_PVT_lenc(1,nyr_PVT_lenc);
init_vector neff_PVT_lenc(1,nyr_PVT_lenc);
init_matrix obs_PVT_lenc(1,nyr_PVT_lenc,1,nlenbins);
init_int nyr_PVT_lenc_pool;     //years and weights to pool predicted PVC length comps to match pooled observations
init_ivector yrs_PVT_lenc_pool(1,nyr_PVT_lenc_pool);
init_vector nsamp_PVT_lenc_pool(1,nyr_PVT_lenc_pool);
// Age Compositions
```

```
init_int nyr_PVT_agec;
init_ivector yrs_PVT_agec(1,nyr_PVT_agec);
init_vector nsamp_PVT_agec(1,nyr_PVT_agec);
init_vector neff_PVT_agec(1,nyr_PVT_agec);
init_matrix obs_PVT_agec(1,nyr_PVT_agec,1,nages);

//##################Parameter values and initial guesses #################################
//#############################################################################
//Discard mortality constants
init_number set_Dmort_cL;
init_number set_Dmort_HB;
init_number set_Dmort_PVT;

// 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_sd;
init_number set_len_sd_se;
//TL(mm)-weight(whole weight in g) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;
//weight(whole weight)-gonad weight (units=g) relationship: GW=aW^b
init_number gwgtpar_a;
init_number gwgtpar_b;
//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age

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


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



//Initial guesses or fixed values of estimated selectivity parameters

//init_number set_selpar_L50_cL1;
//init_number set_selpar_slope_cL1;
//init_number set_selpar_L502_cL1;
//init_number set_selpar_slope2_cL1;
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_L502_cL;
init_number set_selpar_slope2_cL;
init_number set_selpar_min_cL;
init_int    set_selpar_afull_cL;

init_number set_selpar_Age1_cL_D3;
init_number set_selpar_Age2_cL_D3;


init_number set_selpar_L50_cD2;
init_number set_selpar_slope_cD2;
init_number set_selpar_L50_cD3;
init_number set_selpar_slope_cD3;
init_number set_selpar_L502_cD;
init_number set_selpar_slope2_cD;
init_number set_selpar_min_cD;
init_int    set_selpar_afull_cD;

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_L502_HB;
init_number set_selpar_slope2_HB;
init_number set_selpar_min_HB;
init_int    set_selpar_afull_HB;

init_number set_selpar_Age1_HB_D3;

init_number set_selpar_L50_PVT2;
init_number set_selpar_slope_PVT2;
init_number set_selpar_L50_PVT3;
init_number set_selpar_slope_PVT3;
init_number set_selpar_L502_PVT;
init_number set_selpar_slope2_PVT;
init_number set_selpar_min_PVT;
init_int    set_selpar_afull_PVT;

init_number set_sel_initial_wgt_cL; //weight of comm to initial sel (based on initial mean landings in knum)
init_number set_sel_initial_wgt_HB; //weight of for-hire to initial sel (based on initial mean landings in knum)
```

```
init_number set_sel_initial_wgt_PVT;//weight of pvt to initial sel (based on initial mean landings in knum)

//--weights for likelihood components--------------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_D;
init_number set_w_lc_cL;
init_number set_w_lc_cL_D;
init_number set_w_lc_cD;
init_number set_w_lc_HB;
init_number set_w_lc_HB_D;
init_number set_w_lc_PVT;
init_number set_w_ac_cL;
init_number set_w_ac_cD;
init_number set_w_ac_HB;
init_number set_w_ac_PVT;
init_number set_w_I_HBD;
init_number set_w_I_cL;
init_number set_w_I_HB;
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

//Initial guess for recreational for-hire and pvt historic landings multiplicative bias
init_number set_L_hb_bias;
init_number set_L_pvt_bias;

//////--index catchability--------------------------------------------------------------------------------------------------------------
init_number set_logq_HBD;     //catchability coefficient (log) for HBD
init_number set_logq_cL;      //catchability coefficient (log) for commercial logbook CPUE index
init_number set_logq_HB;      //catchability coefficient (log) for the headboat index

//rate of increase on q
init_int set_q_rate_phase;  //value sets estimation phase of rate increase, negative value turns it off
init_number set_q_rate;
//density dependence on fishery q's
init_int set_q_DD_phase;      //value sets estimation phase of random walk, negative value turns it off
init_number set_q_DD_beta;    //value of 0.0 is density 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_HBD_var;    //assumed variance of RW q
init_number set_q_RW_cL_var;     //assumed variance of RW q
init_number set_q_RW_HB_var;     //assumed variance of RW q


////--F's--------------------------------
init_number set_log_avg_F_cL;
init_number set_log_avg_F_cD;
init_number set_log_avg_F_HB;
init_number set_log_avg_F_PVT;
init_number set_F_init;        //initial F, scaled by F_init_ratio
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_cL_D;
init_number set_log_avg_F_HB_D;
init_number set_log_avg_F_PVT_D;

//Multiplicative adjustment to CVs on landings and discards (applied to all fleets and all years)
init_number LD_cv_adj;

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


//threshold sample sizes for length comps
init_number minSS_cL_lenc;
init_number minSS_cL_D_lenc;
init_number minSS_cD_lenc;
init_number minSS_HB_lenc;
init_number minSS_HB_D_lenc;
init_number minSS_PVT_lenc;

//threshold sample sizes for age comps
init_number minSS_cL_agec;
init_number minSS_cD_agec;
init_number minSS_HB_agec;
init_number minSS_PVT_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_cD2;
init_number set_p_lenc_cD3;
init_number set_p_lenc_HB2;
init_number set_p_lenc_HB3;
init_number set_p_lenc_PVT2;
init_number set_p_lenc_PVT3;

init_number set_p_lenc_cL_D3;
init_number set_p_lenc_HB_D2;
init_number set_p_lenc_HB_D3;
init_number set_p_lenc_PVT_D2;
```

```
init_number set_p_lenc_PVT_D3;

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

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

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(500,1100,2);
  //init_bounded_number K(0.05,0.5,2);
  //init_bounded_number t0(-1.0,0.0,2);
  number Linf;
  number K;
  number t0;
  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 gonad_wgt_mt(1,nages); //gonad wgt in mt

  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_cD_mm(styr,endyr,1,nages);       //mean length at age of cD landings in mm (may differ from popn mean)
  matrix wgt_cD_klb(styr,endyr,1,nages);      //whole wgt of cD 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_PVT_mm(styr,endyr,1,nages);      //mean length at age of PVT landings in mm (may differ from popn mean)
  matrix wgt_PVT_klb(styr,endyr,1,nages);     //whole wgt of PVT landings in 1000 lb

  matrix len_cL_D_mm(styr,endyr,1,nages);     //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_cL_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_PVT_D_mm(styr,endyr,1,nages);    //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_PVT_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, 3 cm bins) in population
  matrix lenprob_plus(1,nages,1,nlenbins_plus); //used to compute mass in last length bin (a plus group)
  matrix lenprob_all(1,nages,1,nlenbins_all);   //extended lenprob
  vector lenbins_all(1,nlenbins_all);

  //matrices below are used to match age comps
  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_cD2(1,nages,1,nlenbins);     //distn of size at age in cD block 2
  matrix lenprob_cD3(1,nages,1,nlenbins);     //distn of size at age in cD block 3
  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_PVT1(1,nages,1,nlenbins);    //distn of size at age in PVT block 2
  matrix lenprob_PVT2(1,nages,1,nlenbins);    //distn of size at age in PVT block 2
  matrix lenprob_PVT3(1,nages,1,nlenbins);    //distn of size at age in PVT block 3

  matrix lenprob_cL_D3(1,nages,1,nlenbins);     //distn of size at age in cL 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_PVT_D2(1,nages,1,nlenbins);    //distn of size at age in HB discards rec block 2
  matrix lenprob_PVT_D3(1,nages,1,nlenbins);    //distn of size at age in HB discards rec block 3

  //matrices below pertain to the popn at large, 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_cD2_all(1,nages,1,nlenbins_all);     //distn of size at age in cD block 2
  matrix lenprob_cD3_all(1,nages,1,nlenbins_all);     //distn of size at age in cD block 3
  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_PVT1_all(1,nages,1,nlenbins_all);    //distn of size at age in PVT block 2
  matrix lenprob_PVT2_all(1,nages,1,nlenbins_all);    //distn of size at age in PVT block 2
  matrix lenprob_PVT3_all(1,nages,1,nlenbins_all);    //distn of size at age in PVT block 3

  matrix lenprob_cL_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_PVT_D2_all(1,nages,1,nlenbins_all);    //distn of size at age in HB discards rec block 2
  matrix lenprob_PVT_D3_all(1,nages,1,nlenbins_all);    //distn of size at age in HB discards rec block 3

  //set min and max equal for constant sd or cv
  init_bounded_number len_sd_val(30.0,150.0,4);
//  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
//  number log_len_cv
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

////----Predicted length and age compositions
  matrix pred_cL_lenc(1,nyr_cL_lenc,1,nlenbins);
  matrix pred_cL_D_lenc(1,nyr_cL_D_lenc,1,nlenbins);
  matrix pred_cD_lenc(1,nyr_cD_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_PVT_lenc(1,nyr_PVT_lenc,1,nlenbins);
  matrix L_PVT_num_pool(1,nyr_PVT_lenc,1,nages);          //landings (numbers) at age pooled for length comps
  matrix L_PVT_num_pool_yr(1,nyr_PVT_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
//  //init_bounded_number p_lenc_cL(0.0,1.0,3);
//  //init_bounded_number p_lenc_cD(0.0,1.0,3);
//  //init_bounded_number p_lenc_HB2(0.0,1.0,3);
//  //init_bounded_number p_lenc_HB3(0.0,1.0,3);
//  //init_bounded_number p_lenc_PVT2(0.0,1.0,3);
//  //init_bounded_number p_lenc_PVT3(0.0,1.0,3);
  number p_lenc_cL2;
  number p_lenc_cL3;
  number p_lenc_cD2;
  number p_lenc_cD3;
  number p_lenc_HB2;
  number p_lenc_HB3;
  number p_lenc_PVT2;
  number p_lenc_PVT3;

  //init_bounded_number p_lenc_HB_D3(0.0,1.0,3);
  number p_lenc_cL_D3;
  number p_lenc_HB_D2;  //no comp data in this period, this par only used for avg weight
  number p_lenc_HB_D3;
  number p_lenc_PVT_D2; //no comp data in this period, this par only used for avg weight
  number p_lenc_PVT_D3;

  matrix pred_cL_agec(1,nyr_cL_agec,1,nages);
  matrix ErrorFree_cL_agec(1,nyr_cL_agec,1,nages);
  matrix pred_cD_agec(1,nyr_cD_agec,1,nages);
  matrix ErrorFree_cD_agec(1,nyr_cD_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_PVT_agec(1,nyr_PVT_agec,1,nages);
  matrix ErrorFree_PVT_agec(1,nyr_PVT_agec,1,nages);

  //nsamp_X_allyr vectors used only for R output of comps with nonconsecutive yrs, given sample size cutoffs
  vector nsamp_cL_lenc_allyr(styr,endyr);
  vector nsamp_cL_D_lenc_allyr(styr,endyr);
  vector nsamp_cD_lenc_allyr(styr,endyr);
  vector nsamp_HB_lenc_allyr(styr,endyr);
  vector nsamp_HB_D_lenc_allyr(styr,endyr);
  vector nsamp_PVT_lenc_allyr(styr,endyr);
  vector nsamp_cL_agec_allyr(styr,endyr);
  vector nsamp_cD_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);
  vector nsamp_PVT_agec_allyr(styr,endyr);

//effective sample size applied in multinomial distributions
  vector neff_cL_lenc_allyr(styr,endyr);
  vector neff_cL_D_lenc_allyr(styr,endyr);
  vector neff_cD_lenc_allyr(styr,endyr);
  vector neff_HB_lenc_allyr(styr,endyr);
  vector neff_HB_D_lenc_allyr(styr,endyr);
  vector neff_PVT_lenc_allyr(styr,endyr);
  vector neff_cL_agec_allyr(styr,endyr);
  vector neff_cD_agec_allyr(styr,endyr);
  vector neff_HB_agec_allyr(styr,endyr);
  vector neff_PVT_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_cL_lenc_allyr_out(styr,endyr);
  vector neff_cL_D_lenc_allyr_out(styr,endyr);
  vector neff_cD_lenc_allyr_out(styr,endyr);
  vector neff_HB_lenc_allyr_out(styr,endyr);
  vector neff_HB_D_lenc_allyr_out(styr,endyr);
  vector neff_PVT_lenc_allyr_out(styr,endyr);
  vector neff_cL_agec_allyr_out(styr,endyr);
  vector neff_cD_agec_allyr_out(styr,endyr);
  vector neff_HB_agec_allyr_out(styr,endyr);
  vector neff_PVT_agec_allyr_out(styr,endyr);


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

////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(11,16,1);        //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_logL_add;                        //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr,-3,3,2);  //log recruitment deviations
  init_bounded_vector log_rec_historic_dev(styr+1,styr_rec_dev-1,-3,3,2); //log recruitment deviations early period
  //vector log_rec_dev(styr_rec_dev,endyr);
  vector log_rec_dev_output(styr,endyr+1);           //used in output. equals zero except for yrs in log_rec_dev
  vector log_rec_historic_dev_output(styr,endyr+1);  //used in output. equals zero except for yrs in log_rec_dev

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

  number BiasCor;                               //Bias correction in equilibrium recruits
  init_bounded_number R_autocorr(-1.0,1.0,-1);  //autocorrelation in SR
  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-------------------------------------------------------------------------

//Commercial handline-------------------------------------------------
  matrix sel_cL(styr,endyr,1,nages);
  ////init_bounded_number selpar_L50_cL1(0.1,8.0,1);
  ////init_bounded_number selpar_slope_cL1(0.5,12.0,1); //period 1
  //number selpar_slope_cL1; //period 1
  //number selpar_L50_cL1;
  //number selpar_slope2_cL1; //period 1
  //number selpar_L502_cL1;

  init_bounded_number selpar_L50_cL2(0.1,8.0,1);
  init_bounded_number selpar_slope_cL2(0.5,12.0,1); //period 2

  init_bounded_number selpar_L50_cL3(0.1,8.0,1);
  init_bounded_number selpar_slope_cL3(0.5,12.0,1); //period 3
  //number selpar_slope_cL3; //period 3
  //number selpar_L50_cL3;
  //number selpar_slope2_cL; //period 3
  //number selpar_L502_cL;
  init_bounded_number selpar_L502_cL(4.0,15.0,-3);   //period 3
  init_bounded_number selpar_slope2_cL(0.1,5.0,-3);
  init_bounded_number selpar_min_cL(0.0,1.0,-3);

  //init_bounded_dev_vector selpar_L50_cL_dev(styr_cL_lenc,endyr_period1,-5,5,3);
  //vector sel_cL_1(1,nages); //sel in period 1
  vector sel_cL_2(1,nages); //sel in period 2
  vector sel_cL_3(1,nages); //sel in period 3

//Commercial handline Discards-------------------------------------------------
  matrix sel_cL_D(styr,endyr,1,nages); //selectivity assumed same
  vector vecprob_cL_D3(4,nages);       //prob of less than size limit
  init_bounded_number selpar_Age1_cL_D3_logit(-10.0,10.0,1); //estimated in logit space: period 3
  init_bounded_number selpar_Age2_cL_D3_logit(-10.0,10.0,1); //estimated in logit space: period 3
  number prior_selpar_Age1_cL_D3_logit;                      //prior in logit space
  number prior_selpar_Age2_cL_D3_logit;                      //prior in logit space
  number selpar_Age1_cL_D3; //period 3
  number selpar_Age2_cL_D3; //period 3
  vector sel_cL_D_3(1,nages); //sel in period 3

//Commercial diving gear-------------------------------------------------
  matrix sel_cD(styr,endyr,1,nages);

  number selpar_L50_cD2;
  number selpar_slope_cD2;

  init_bounded_number selpar_L50_cD3(0.1,8.0,1);
  init_bounded_number selpar_slope_cD3(0.5,12.0,1);
  //number selpar_L50_cD3;
  //number selpar_slope_cD3;
  //number selpar_L502_cD;
  //number selpar_slope2_cD;
  init_bounded_number selpar_L502_cD(4.0,15.0,3);   //period 3
  init_bounded_number selpar_slope2_cD(0.1,5.0,3);
  init_bounded_number selpar_min_cD(0.0,1.0,3);

//  vector sel_cD_1(1,nages); //sel vector
  vector sel_cD_2(1,nages);   //sel vector
  vector sel_cD_3(1,nages);   //sel vector

//Headboat-------------------------------------------------
  matrix sel_HB(styr,endyr,1,nages);
  init_bounded_number selpar_L50_HB1(0.1,6.0,1);
```

```
  init_bounded_number selpar_slope_HB1(0.5,12.0,1); //period 1
  //number selpar_slope_HB1; //period 1
  //number selpar_L50_HB1;

  init_bounded_number selpar_L50_HB2(0.1,6.0,1);
  init_bounded_number selpar_slope_HB2(0.5,12.0,1); //period 2
  //number selpar_slope_HB2; //period 2
  //number selpar_L50_HB2;

  init_bounded_number selpar_L50_HB3(0.1,8.0,1);
  init_bounded_number selpar_slope_HB3(0.5,12.0,1); //period 3
  //number selpar_slope_HB3; //period 3
  //number selpar_L50_HB3;
  //number selpar_slope2_HB; //period 3
  //number selpar_L502_HB;
  init_bounded_number selpar_L502_HB(2.5,15.0,3);   //period 3
  init_bounded_number selpar_slope2_HB(0.1,5.0,3);
  init_bounded_number selpar_min_HB(0.0,1.0,-3);


//  //init_bounded_dev_vector selpar_L50_HB_dev(styr_HB_lenc,endyr_period1,-5,5,3);
  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

//Headboat  Discards selectivity-------------------------------------------------
  matrix sel_HB_D(styr,endyr,1,nages);
  vector vecprob_HB_D2(3,nages);     //prob of less than size limit
  vector vecprob_HB_D3(3,nages);     //prob of less than size limit

  init_bounded_number selpar_Age1_HB_D3_logit(-10.0,10.0,1); //estimated in logit space: period2, period 3
  number prior_selpar_Age1_HB_D3_logit;                      //prior in logit space
  number selpar_Age1_HB_D3;                                  //period2, period 3
//   number selpar_Age1_HB_D3;

//  init_bounded_number selpar_L50_HB_D3(0.1,8.0,1);
//  init_bounded_number selpar_slope_HB_D3(0.5,12.0,1); //period 3
//  init_bounded_number selpar_slope2_HB_D3(1.0,6.0,3);
//  init_bounded_number selpar_L502_HB_D3(0.0,12.0,3);


//  vector sel_HB_D_1(1,nages); //sel in period 1
  vector sel_HB_D_2(1,nages); //sel in period 2
  vector sel_HB_D_3(1,nages); //sel in period 3

////PVT selectivity-------------------------------------------------
  matrix sel_PVT(styr,endyr,1,nages);

  init_bounded_number selpar_L50_PVT2(0.1,8.0,1);
  init_bounded_number selpar_slope_PVT2(0.5,12.0,1); //period 2
//  number selpar_slope_PVT2; //period 2
//  number selpar_L50_PVT2;

  init_bounded_number selpar_L50_PVT3(0.1,8.0,1);
  init_bounded_number selpar_slope_PVT3(0.5,12.0,1); //period 3
//  number selpar_slope_PVT3; //period 3
//  number selpar_L50_PVT3;
  //number selpar_slope2_PVT; //period 3
  //number selpar_L502_PVT;
  init_bounded_number selpar_L502_PVT(2.5,15.0,-3);   //period 3
  init_bounded_number selpar_slope2_PVT(0.1,12.0,-3);
  init_bounded_number selpar_min_PVT(0.0,1.0,-3);

  //init_bounded_dev_vector selpar_L50_PVT_dev(styr_PVT_lenc,endyr_period1,-5,5,3);
  //vector sel_PVT_1(1,nages); //sel in period 1
  vector sel_PVT_2(1,nages); //sel in period 2
  vector sel_PVT_3(1,nages); //sel in period 3
  //PVT discard sel
  matrix sel_PVT_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_HBD_cpue(styr_HBD_cpue,endyr_HBD_cpue);          //predicted HBD U (fish/trap-hour)
  matrix N_HBD(styr_HBD_cpue,endyr_HBD_cpue,1,nages);          //used to compute HBD 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 (number/angler-day)
  matrix N_HB(styr_HB_cpue,endyr_HB_cpue,1,nages);             //used to compute HB index

//---Catchability (CPUE q's)----------------------------------------------------------
  init_bounded_number log_q_cL(-15,-5,1);
  init_bounded_number log_q_HB(-20,-5,1);
  init_bounded_number log_q_HBD(-20,-5,1);
  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_pvt_bias(0.1,10.0,3);
  number L_hb_bias;
  number L_pvt_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_cD_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cD_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cD_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cD_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

  matrix L_PVT_num(styr,endyr,1,nages);  //landings (numbers) at age
  matrix L_PVT_klb(styr,endyr,1,nages);  //landings (1000 lb whole weight) at age
  vector pred_PVT_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_PVT_L_klb(styr,endyr);     //yearly landings in 1000 lb summed over ages

  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_cL_num(styr,endyr,1,nages);   //discards (numbers) at age
  matrix D_cL_klb(styr,endyr,1,nages);   //discards (1000 lb) at age
  vector pred_cL_D_knum(styr,endyr);     //yearly discards summed over ages
  vector obs_cL_D(styr_cL_D,endyr_cL_D); //observed releases multiplied by discard mortality
  vector pred_cL_D_klb(styr,endyr);      //yearly discards in klb summed over ages

  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 obs_HB_D(styr_HB_D,endyr_HB_D); //observed releases multiplied by discard mortality
  vector pred_HB_D_klb(styr,endyr);      //yearly discards in klb summed over ages

  matrix D_PVT_num(styr,endyr,1,nages);  //discards (numbers) at age
  matrix D_PVT_klb(styr,endyr,1,nages);  //discards (1000 lb) at age
  vector pred_PVT_D_knum(styr,endyr);    //yearly discards summed over ages
  vector obs_PVT_D(styr_PVT_D,endyr_PVT_D); //observed releases multiplied by discard mortality
  vector pred_PVT_D_klb(styr,endyr);     //yearly discards in klb summed over ages

  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_cD_prop;       //proportion of F_sum attributable to diving, last X yrs
  number F_HB_prop;       //proportion of F_sum attributable to headboat, last X yrs
  number F_PVT_prop;      //proportion of F_sum attributable to PVT, last X yrs
  number F_cL_D_prop;     //proportion of F_sum attributable to hal discards, last X yrs
  number F_HB_D_prop;     //proportion of F_sum attributable to headboat discards, last X yrs
  number F_PVT_D_prop;    //proportion of F_sum attributable to PVT 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 (total mature biomass) at msy
  number F_msy_out;             //F at msy
  number msy_klb_out;           //max sustainable yield (1000 lb)
  number msy_knum_out;          //max sustainable yield (1000 fish)
  number B_msy_out;             //total biomass at MSY
  number R_msy_out;             //equilibrium recruitment at F=Fmsy
  number 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_mdyr(1,nages);    //numbers at age for MSY calculations: mdpt of yr
  vector L_age_msy(1,nages);         //catch at age for MSY calculations
  vector Z_age_msy(1,nages);         //total mortality at age for MSY calculations
  vector 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_cD(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cD(styr_cD_L,endyr_cD_L,-10.0,5.0,2);
  matrix F_cD(styr,endyr,1,nages);
  vector F_cD_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_cD;

  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_PVT(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_PVT(styr_PVT_L,endyr_PVT_L,-10.0,5.0,2);
  matrix F_PVT(styr,endyr,1,nages);
  vector F_PVT_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_PVT;
  number log_F_dev_end_PVT;

  init_bounded_number F_init(0.01,0.5,-1);
  number F_init_ratio;
  //number F_init_ratio;   //scales initial F, which is read in as a fixed value
  vector sel_initial(1,nages);       //initial selectivity (a combination of for-hire and commercial selectivities)

//--Discard mortality stuff------------------------------------------------------------------------------
  init_bounded_number log_avg_F_cL_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_cL_D(styr_cL_D,endyr_cL_D,-10.0,5.0,2);
  matrix F_cL_D(styr,endyr,1,nages);
  vector F_cL_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_cL_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_PVT_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_PVT_D(styr_PVT_D,endyr_PVT_D,-10.0,5.0,2);
  matrix F_PVT_D(styr,endyr,1,nages);
  vector F_PVT_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_PVT_D;

  number Dmort_cL;
  number Dmort_HB;
  number Dmort_PVT;

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

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

```
  number bpr_F0;                     //Biomass per recruit at F=0

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


////-------Objective function components-----------------------------------------------------------------------------
  number w_L;
  number w_D;
  number w_lc_cL;
  number w_lc_cL_D;
  number w_lc_cD;
  number w_lc_HB;
  number w_lc_HB_D;
  number w_lc_PVT;
  number w_ac_cL;
  number w_ac_cD;
  number w_ac_HB;
  number w_ac_PVT;
  number w_I_HBD;
  number w_I_cL;
  number w_I_HB;
  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_HBD_cpue;
  number f_cL_cpue;
  number f_HB_cpue;

  number f_cL_L;
  number f_cD_L;
  number f_HB_L;
  number f_PVT_L;

  number f_cL_D;
  number f_HB_D;
  number f_PVT_D;

  number f_cL_lenc;
  number f_cL_D_lenc;
  number f_cD_lenc;
  number f_HB_lenc;
  number f_HB_D_lenc;
  number f_PVT_lenc;

  number f_cL_agec;
  number f_cD_agec;
  number f_HB_agec;
  number f_PVT_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_rec_historic_dev;       //extra weight on deviations in first recruitment stanza
  number f_Ftune;                  //penalty for tuning F in Ftune yr.  Not applied in final optimization phase.
  number f_fullF_constraint;       //penalty for Fapex>X
//  number f_cvlen_dev_constraint; //deviation penalty on cv's of length at age
//  number f_cvlen_diff_constraint;//first diff penalty on cv's of length at age
  number f_priors;                  //prior information on parameters

  //init_number xdum;
  objective_function_value fval;
  number fval_data;

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

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


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

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
RUNTIME_SECTION
 maximum_function_evaluations 1000, 2000,1000, 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
  Dmort_cL=set_Dmort_cL;
  Dmort_HB=set_Dmort_HB;
  Dmort_PVT=set_Dmort_PVT;

  obs_cL_D=Dmort_cL*obs_cL_released;
  obs_HB_D=Dmort_HB*obs_HB_released;
```

```
  obs_PVT_D=Dmort_PVT*obs_PVT_released;

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


////  age_limit_12in=t0-log(1.0-limit_12in/Linf)/K; //age at size limit: 12" limit;
////  age_limit_20in=t0-log(1.0-limit_20in/Linf)/K; //age at size limit: 20" 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_HBD=set_logq_HBD;
  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_pvt_bias=set_L_pvt_bias;

  w_L=set_w_L;
  w_D=set_w_D;
  w_lc_cL=set_w_lc_cL;
  w_lc_cL_D=set_w_lc_cL_D;
  w_lc_cD=set_w_lc_cD;
  w_lc_HB=set_w_lc_HB;
  w_lc_HB_D=set_w_lc_HB_D;
  w_lc_PVT=set_w_lc_PVT;
  w_ac_cL=set_w_ac_cL;
  w_ac_cD=set_w_ac_cD;
  w_ac_HB=set_w_ac_HB;
  w_ac_PVT=set_w_ac_PVT;
  w_I_HBD=set_w_I_HBD;
  w_I_cL=set_w_I_cL;
  w_I_HB=set_w_I_HB;
  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_cD=set_log_avg_F_cD;
  log_avg_F_HB=set_log_avg_F_HB;
  log_avg_F_PVT=set_log_avg_F_PVT;
  F_init_ratio=set_F_init_ratio;
  F_init=set_F_init;

  log_avg_F_cL_D=set_log_avg_F_cL_D;
  log_avg_F_HB_D=set_log_avg_F_HB_D;
  log_avg_F_PVT_D=set_log_avg_F_PVT_D;

  len_sd_val=set_len_sd;

  log_R0=set_log_R0;


  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_L502_cL=set_selpar_L502_cL;
  selpar_slope2_cL=set_selpar_slope2_cL;
  selpar_min_cL=set_selpar_min_cL;
  selpar_afull_cL=set_selpar_afull_cL;

  selpar_L50_cD2=set_selpar_L50_cD2;
  selpar_slope_cD2=set_selpar_slope_cD2;
  selpar_L50_cD3=set_selpar_L50_cD3;
  selpar_L502_cD=set_selpar_L502_cD;
  selpar_slope_cD3=set_selpar_slope_cD3;
  selpar_slope2_cD=set_selpar_slope2_cD;
  selpar_min_cD=set_selpar_min_cD;
  selpar_afull_cD=set_selpar_afull_cD;

  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_L502_HB=set_selpar_L502_HB;
  selpar_slope2_HB=set_selpar_slope2_HB;
  selpar_min_HB=set_selpar_min_HB;
  selpar_afull_HB=set_selpar_afull_HB;

  prior_selpar_Age1_HB_D3_logit=-log((1.0-set_selpar_Age1_HB_D3)/set_selpar_Age1_HB_D3); //inverse of logit
  prior_selpar_Age1_cL_D3_logit=-log((1.0-set_selpar_Age1_cL_D3)/set_selpar_Age1_cL_D3); //inverse of logit
  prior_selpar_Age2_cL_D3_logit=-log((1.0-set_selpar_Age2_cL_D3)/set_selpar_Age2_cL_D3); //inverse of logit

  selpar_Age1_HB_D3_logit=prior_selpar_Age1_HB_D3_logit; //inverse of logit
  selpar_Age1_cL_D3_logit=prior_selpar_Age1_cL_D3_logit; //inverse of logit
  selpar_Age2_cL_D3_logit=prior_selpar_Age2_cL_D3_logit; //inverse of logit

//  selpar_L50_HB_D3=set_selpar_L50_HB_D3;
//  selpar_slope_HB_D3=set_selpar_slope_HB_D3; //period 3
//  selpar_slope2_HB_D3=set_selpar_slope2_HB_D3; //period 3
//  selpar_L502_HB_D3=set_selpar_L502_HB_D3;

  selpar_L50_PVT2=set_selpar_L50_PVT2;
  selpar_slope_PVT2=set_selpar_slope_PVT2;
  selpar_L50_PVT3=set_selpar_L50_PVT3;
  selpar_slope_PVT3=set_selpar_slope_PVT3;
  selpar_L502_PVT=set_selpar_L502_PVT;
  selpar_slope2_PVT=set_selpar_slope2_PVT;
  selpar_min_PVT=set_selpar_min_PVT;
  selpar_afull_PVT=set_selpar_afull_PVT;


 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;           //additive constant to prevent division by zero

 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_cD2=set_p_lenc_cD2;
 p_lenc_cD3=set_p_lenc_cD3;
 p_lenc_HB2=set_p_lenc_HB2;
 p_lenc_HB3=set_p_lenc_HB3;
 p_lenc_PVT2=set_p_lenc_PVT2;
 p_lenc_PVT3=set_p_lenc_PVT3;

 p_lenc_cL_D3=set_p_lenc_cL_D3;
 p_lenc_HB_D2=set_p_lenc_HB_D2;
 p_lenc_HB_D3=set_p_lenc_HB_D3;
 p_lenc_PVT_D2=set_p_lenc_PVT_D2;
 p_lenc_PVT_D3=set_p_lenc_PVT_D3;

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

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

      nsamp_cL_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nsamp_cL_D_lenc_allyr=missing;
      nsamp_cD_lenc_allyr=missing;
      nsamp_HB_lenc_allyr=missing;
      nsamp_HB_D_lenc_allyr=missing;
      nsamp_PVT_lenc_allyr=missing;
      nsamp_cL_agec_allyr=missing;
      nsamp_cD_agec_allyr=missing;
      nsamp_HB_agec_allyr=missing;
      nsamp_PVT_agec_allyr=missing;

      neff_cL_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      neff_cL_D_lenc_allyr=missing;
      neff_cD_lenc_allyr=missing;
      neff_HB_lenc_allyr=missing;
      neff_HB_D_lenc_allyr=missing;
      neff_PVT_lenc_allyr=missing;
      neff_cL_agec_allyr=missing;
```

```
      neff_cD_agec_allyr=missing;
      neff_HB_agec_allyr=missing;
      neff_PVT_agec_allyr=missing;


      for (iyear=1; iyear<=nyr_cL_lenc; iyear++)
         {if (nsamp_cL_lenc(iyear)>=minSS_cL_lenc)
           {nsamp_cL_lenc_allyr(yrs_cL_lenc(iyear))=nsamp_cL_lenc(iyear);
             neff_cL_lenc_allyr(yrs_cL_lenc(iyear))=neff_cL_lenc(iyear);
           }}

      for (iyear=1; iyear<=nyr_cL_D_lenc; iyear++)
         {if (nsamp_cL_D_lenc(iyear)>=minSS_cL_D_lenc)
           {nsamp_cL_D_lenc_allyr(yrs_cL_D_lenc(iyear))=nsamp_cL_D_lenc(iyear);
             neff_cL_D_lenc_allyr(yrs_cL_D_lenc(iyear))=neff_cL_D_lenc(iyear);
         }}

      for (iyear=1; iyear<=nyr_cD_lenc; iyear++)
         {if (nsamp_cD_lenc(iyear)>=minSS_cD_lenc)
           {nsamp_cD_lenc_allyr(yrs_cD_lenc(iyear))=nsamp_cD_lenc(iyear);
             neff_cD_lenc_allyr(yrs_cD_lenc(iyear))=neff_cD_lenc(iyear);
         }}

      for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
         {if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
           {nsamp_HB_lenc_allyr(yrs_HB_lenc(iyear))=nsamp_HB_lenc(iyear);
             neff_HB_lenc_allyr(yrs_HB_lenc(iyear))=neff_HB_lenc(iyear);
         }}

      for (iyear=1; iyear<=nyr_HB_D_lenc; iyear++)
         {if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
           {nsamp_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=nsamp_HB_D_lenc(iyear);
             neff_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=neff_HB_D_lenc(iyear);
         }}

      for (iyear=1; iyear<=nyr_PVT_lenc; iyear++)
         {if (nsamp_PVT_lenc(iyear)>=minSS_PVT_lenc)
           {nsamp_PVT_lenc_allyr(yrs_PVT_lenc(iyear))=nsamp_PVT_lenc(iyear);
             neff_PVT_lenc_allyr(yrs_PVT_lenc(iyear))=neff_PVT_lenc(iyear);
         }}

      for (iyear=1; iyear<=nyr_cL_agec; iyear++)
         {if (nsamp_cL_agec(iyear)>=minSS_cL_agec)
           {nsamp_cL_agec_allyr(yrs_cL_agec(iyear))=nsamp_cL_agec(iyear);
            neff_cL_agec_allyr(yrs_cL_agec(iyear))=neff_cL_agec(iyear);
         }}
      for (iyear=1; iyear<=nyr_cD_agec; iyear++)
         {if (nsamp_cD_agec(iyear)>=minSS_cD_agec)
           {nsamp_cD_agec_allyr(yrs_cD_agec(iyear))=nsamp_cD_agec(iyear);
            neff_cD_agec_allyr(yrs_cD_agec(iyear))=neff_cD_agec(iyear);
         }}
      for (iyear=1; iyear<=nyr_HB_agec; iyear++)
         {if (nsamp_HB_agec_allyr(iyear)>=minSS_HB_agec)
           {nsamp_HB_agec_allyr(yrs_HB_agec(iyear))=nsamp_HB_agec(iyear);
            neff_HB_agec_allyr(yrs_HB_agec(iyear))=neff_HB_agec(iyear);
         }}
      for (iyear=1; iyear<=nyr_PVT_agec; iyear++)
         {if (nsamp_PVT_agec(iyear)>=minSS_PVT_agec)
           {nsamp_PVT_agec_allyr(yrs_PVT_agec(iyear))=nsamp_PVT_agec(iyear);
            neff_PVT_agec_allyr(yrs_PVT_agec(iyear))=neff_PVT_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();
  L_cL_num.initialize();
  F_cD.initialize();
  L_cD_num.initialize();
  F_HB.initialize();
  L_HB_num.initialize();
  F_PVT.initialize();
  L_PVT_num.initialize();

  F_cL_out.initialize();
  F_cD_out.initialize();
  F_HB_out.initialize();
  F_PVT_out.initialize();

  F_cL_D_out.initialize();
  F_HB_D_out.initialize();
  F_PVT_D_out.initialize();

  F_cL_D.initialize();
  D_cL_num.initialize();
  pred_cL_D_klb.initialize();
  F_HB_D.initialize();
  D_HB_num.initialize();
  pred_HB_D_klb.initialize();
  F_PVT_D.initialize();
  D_PVT_num.initialize();
  pred_PVT_D_klb.initialize();
```

```
  sel_cL.initialize();
  sel_cD.initialize();
  sel_HB.initialize();
  sel_PVT.initialize();
  sel_cL_D.initialize();
  sel_HB_D.initialize();
  sel_PVT_D.initialize();


  log_rec_dev_output.initialize();
  log_rec_historic_dev_output.initialize();
  log_Nage_dev_output.initialize();
  log_rec_historic_dev.initialize();
  log_rec_dev.initialize();
  log_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
    gonad_wgt_mt=g2mt*gwgtpar_a*pow(wgt_g,gwgtpar_b);    //gonad wgt in mt


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

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

  for (iage=1;iage<=nages;iage++)
  {
    //len_cv(iage)=mfexp(log_len_cv+log_len_cv_dev(iage));
    len_sd(iage)=len_sd_val;
    //len_cv(iage)=len_cv_max-(len_cv_max-len_sd)/(1.0+mfexp(-len_cv_slope*(iage-len_cv_a50)));
```

```
  for (ilen=1;ilen<=nlenbins_all;ilen++)
   { lenprob_all(iage,ilen)=(mfexp(-(square(lenbins_all(ilen)-meanlen_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 specific length probs, assumed normal prior to size limits

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_cD2_all=lenprob_all;    //values may be adjusted based on size limit
lenprob_cD3_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_PVT1=lenprob;
lenprob_PVT2_all=lenprob_all;   //values may be adjusted based on size limit
lenprob_PVT3_all=lenprob_all;   //values may be adjusted based on size limit

lenprob_cL_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_PVT_D2_all=lenprob_all; //values may be adjusted based on size limit
lenprob_PVT_D3_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_12in)  //Landings block two
      {
        lenprob_cL2_all(iage,ilen)=p_lenc_cL2*lenprob_all(iage,ilen);
        lenprob_cD2_all(iage,ilen)=p_lenc_cD2*lenprob_all(iage,ilen);
        lenprob_HB2_all(iage,ilen)=p_lenc_HB2*lenprob_all(iage,ilen);
        lenprob_PVT2_all(iage,ilen)=p_lenc_PVT2*lenprob_all(iage,ilen);

      }
      if (lenbins_all(ilen) < limit_20in) //Landings block three
      {
        lenprob_cL3_all(iage,ilen)=p_lenc_cL3*lenprob_all(iage,ilen);
        lenprob_cD3_all(iage,ilen)=p_lenc_cD3*lenprob_all(iage,ilen);
        lenprob_HB3_all(iage,ilen)=p_lenc_HB3*lenprob_all(iage,ilen);
        lenprob_PVT3_all(iage,ilen)=p_lenc_PVT3*lenprob_all(iage,ilen);
       }


      if (lenbins_all(ilen) > limit_disc) //Discards block two
      { lenprob_HB_D2_all(iage,ilen)=p_lenc_HB_D2*lenprob_all(iage,ilen);
        lenprob_PVT_D2_all(iage,ilen)=p_lenc_PVT_D2*lenprob_all(iage,ilen);
      }


      if (lenbins_all(ilen) > limit_20in) //Discards block three
      { lenprob_cL_D3_all(iage,ilen)=p_lenc_cL_D3*lenprob_all(iage,ilen);
        lenprob_HB_D3_all(iage,ilen)=p_lenc_HB_D3*lenprob_all(iage,ilen);
        lenprob_PVT_D3_all(iage,ilen)=p_lenc_PVT_D3*lenprob_all(iage,ilen);
      }

  }  //end ilen loop


  if (iage>=3)  //compute prior to standardizing
      {vecprob_HB_D2(iage)=sum(lenprob_HB_D2_all(iage));
       vecprob_HB_D3(iage)=sum(lenprob_HB_D3_all(iage));}
  if (iage>=4)  //compute prior to standardizing
      {vecprob_cL_D3(iage)=sum(lenprob_cL_D3_all(iage));}

  lenprob_cL2_all(iage)/=sum(lenprob_cL2_all(iage));       //standardize
  lenprob_cL3_all(iage)/=sum(lenprob_cL3_all(iage));       //standardize
  lenprob_cD2_all(iage)/=sum(lenprob_cD2_all(iage));       //standardize
  lenprob_cD3_all(iage)/=sum(lenprob_cD3_all(iage));       //standardize
  lenprob_HB2_all(iage)/=sum(lenprob_HB2_all(iage));       //standardize
  lenprob_HB3_all(iage)/=sum(lenprob_HB3_all(iage));       //standardize
  lenprob_PVT2_all(iage)/=sum(lenprob_PVT2_all(iage));     //standardize
  lenprob_PVT3_all(iage)/=sum(lenprob_PVT3_all(iage));     //standardize

  lenprob_cL_D3_all(iage)/=sum(lenprob_cL_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_PVT_D2_all(iage)/=sum(lenprob_PVT_D2_all(iage)); //standardize
  lenprob_PVT_D3_all(iage)/=sum(lenprob_PVT_D3_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_cD2(iage,ilen)=lenprob_cD2_all(iage,ilen);
     lenprob_cD3(iage,ilen)=lenprob_cD3_all(iage,ilen);
     lenprob_HB2(iage,ilen)=lenprob_HB2_all(iage,ilen);
     lenprob_HB3(iage,ilen)=lenprob_HB3_all(iage,ilen);
     lenprob_PVT2(iage,ilen)=lenprob_PVT2_all(iage,ilen);
     lenprob_PVT3(iage,ilen)=lenprob_PVT3_all(iage,ilen);
     lenprob_cL_D3(iage,ilen)=lenprob_cL_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_PVT_D2(iage,ilen)=lenprob_PVT_D2_all(iage,ilen);
     lenprob_PVT_D3(iage,ilen)=lenprob_PVT_D3_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_cD2(iage)(nlenbins)=lenprob_cD2(iage)(nlenbins)+lenprob_cD2_all(iage)(ilen);
      lenprob_cD3(iage)(nlenbins)=lenprob_cD3(iage)(nlenbins)+lenprob_cD3_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_PVT2(iage)(nlenbins)=lenprob_PVT2(iage)(nlenbins)+lenprob_PVT2_all(iage)(ilen);
      lenprob_PVT3(iage)(nlenbins)=lenprob_PVT3(iage)(nlenbins)+lenprob_PVT3_all(iage)(ilen);
      lenprob_cL_D3(iage)(nlenbins)=lenprob_cL_D3(iage)(nlenbins)+lenprob_cL_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_PVT_D2(iage)(nlenbins)=lenprob_PVT_D2(iage)(nlenbins)+lenprob_PVT_D2_all(iage)(ilen);
      lenprob_PVT_D3(iage)(nlenbins)=lenprob_PVT_D3(iage)(nlenbins)+lenprob_PVT_D3_all(iage)(ilen);
      }

  } //end iage loop

FUNCTION get_weight_at_age_landings

  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
    len_cL_mm(iyear)=meanlen_TL;
    wgt_cL_klb(iyear)=wgt_klb;
//    len_cD_mm(iyear)=meanlen_TL;  //no diving in first block
//    wgt_cD_klb(iyear)=wgt_klb;
    len_HB_mm(iyear)=meanlen_TL;
    wgt_HB_klb(iyear)=wgt_klb;
    len_PVT_mm(iyear)=meanlen_TL;
    wgt_PVT_klb(iyear)=wgt_klb;

    //NO discards in first block
  } // 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_cD_mm(iyear,iage)=sum(elem_prod(lenprob_cD2_all(iage),lenbins_all));
    len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB2_all(iage),lenbins_all));
    len_PVT_mm(iyear,iage)=sum(elem_prod(lenprob_PVT2_all(iage),lenbins_all));
//    len_cL_D_mm(iyear,iage)=sum(elem_prod(lenprob_cL_D2_all(iage),lenbins_all));
    len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D2_all(iage),lenbins_all));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    wgt_cD_klb(iyear)=g2klb*wgtpar_a*pow(len_cD_mm(iyear),wgtpar_b);
    wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_PVT_klb(iyear)=g2klb*wgtpar_a*pow(len_PVT_mm(iyear),wgtpar_b);
//    wgt_cL_D_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_D_mm(iyear),wgtpar_b);
    wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
  }

  for (iyear=(endyr_period2+1); iyear<=endyr; iyear++)
  {
    for (iage=1;iage<=nages; iage++)
    {
    len_cL_mm(iyear,iage)=sum(elem_prod(lenprob_cL3_all(iage),lenbins_all));
    len_cD_mm(iyear,iage)=sum(elem_prod(lenprob_cD3_all(iage),lenbins_all));
    len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB3_all(iage),lenbins_all));
    len_PVT_mm(iyear,iage)=sum(elem_prod(lenprob_PVT3_all(iage),lenbins_all));
    len_cL_D_mm(iyear,iage)=sum(elem_prod(lenprob_cL_D3_all(iage),lenbins_all));
    len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D3_all(iage),lenbins_all));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    wgt_cD_klb(iyear)=g2klb*wgtpar_a*pow(len_cD_mm(iyear),wgtpar_b);
    wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_PVT_klb(iyear)=g2klb*wgtpar_a*pow(len_PVT_mm(iyear),wgtpar_b);
    wgt_cL_D_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_D_mm(iyear),wgtpar_b);
    wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
  }

  //PVT_D assumed same as HB_D
  len_PVT_D_mm=len_HB_D_mm;
  wgt_PVT_D_klb=wgt_HB_D_klb;


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

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


FUNCTION get_selectivity

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

  //---flat-topped sels---------------------------
  for (iage=1; iage<=nages; iage++)
  {
```

```
////     sel_cL_2(iage)=(1./(1.+mfexp(-1.*selpar_slope_cL2*(double(agebins(iage))-
////                       selpar_L50_cL2))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cL2*
////                       (double(agebins(iage))-(selpar_L50_cL2+selpar_L502_cL2)))))); //double logistic

     //commercial lines
     sel_cL_2(iage)=1./(1.+mfexp(-1.*selpar_slope_cL2*(double(agebins(iage))-selpar_L50_cL2))); //logistic
     sel_cL_3(iage)=1./(1.+mfexp(-1.*selpar_slope_cL3*(double(agebins(iage))-selpar_L50_cL3))); //logistic

//     //commercial diving
//      sel_cD_3(iage)=1./(1.+mfexp(-1.*selpar_slope_cD3*(double(agebins(iage))-selpar_L50_cD3)));
//
//     //headboat
//     sel_HB_1(iage)=1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))-selpar_L50_HB1))); //logistic
//     sel_HB_2(iage)=1./(1.+mfexp(-1.*selpar_slope_HB2*(double(agebins(iage))-selpar_L50_HB2))); //logistic
//     sel_HB_3(iage)=1./(1.+mfexp(-1.*selpar_slope_HB3*(double(agebins(iage))-selpar_L50_HB3))); //logistic
//
//     //private
//     sel_PVT_2(iage)=1./(1.+mfexp(-1.*selpar_slope_PVT2*(double(agebins(iage))-selpar_L50_PVT2))); //logistic
//     sel_PVT_3(iage)=1./(1.+mfexp(-1.*selpar_slope_PVT3*(double(agebins(iage))-selpar_L50_PVT3))); //logistic
  }

  //---dome-shaped sels----------------------------

//  sel_cL_2(selpar_afull_cL)=1.0;
//  sel_cL_3(selpar_afull_cL)=1.0;

  sel_cD_3(selpar_afull_cD)=1.0;

  sel_HB_1(selpar_afull_HB)=1.0;
  sel_HB_2(selpar_afull_HB)=1.0;
  sel_HB_3(selpar_afull_HB)=1.0;

  sel_PVT_2(selpar_afull_PVT)=1.0;
  sel_PVT_3(selpar_afull_PVT)=1.0;

  selpar_slope2_PVT=selpar_slope2_HB;
  selpar_L502_PVT=selpar_L502_HB;
  selpar_min_PVT=selpar_min_HB;


  for (iage=1; iage<=nages; iage++)
  {
//   if (iage<selpar_afull_cL) {sel_cL_2(iage)=1./(1.+mfexp(-1.*selpar_slope_cL2*(double(agebins(iage))-selpar_L50_cL2)));
//                                sel_cL_3(iage)=1./(1.+mfexp(-1.*selpar_slope_cL3*(double(agebins(iage))-selpar_L50_cL3)));
//                               }
//   if (iage>selpar_afull_cL){sel_cL_2(iage)=1.0-(1.0-selpar_min_cL)/
//                               (1.+mfexp(-1.*selpar_slope2_cL*(double(agebins(iage))-selpar_L502_cL)));
//                              sel_cL_3(iage)=1.0-(1.0-selpar_min_cL)/
//                               (1.+mfexp(-1.*selpar_slope2_cL*(double(agebins(iage))-selpar_L502_cL)));
//                               }

   if (iage<selpar_afull_cD) {sel_cD_3(iage)=1./(1.+mfexp(-1.*selpar_slope_cD3*(double(agebins(iage))-selpar_L50_cD3)));}
   if (iage>selpar_afull_cD){sel_cD_3(iage)=1.0-(1.0-selpar_min_cD)/
                            (1.+mfexp(-1.*selpar_slope2_cD*(double(agebins(iage))-selpar_L502_cD)));}

   if (iage<selpar_afull_HB) {
                              sel_HB_1(iage)=1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))-selpar_L50_HB1)));
                              sel_HB_2(iage)=1./(1.+mfexp(-1.*selpar_slope_HB2*(double(agebins(iage))-selpar_L50_HB2)));
                              sel_HB_3(iage)=1./(1.+mfexp(-1.*selpar_slope_HB3*(double(agebins(iage))-selpar_L50_HB3)));
                             }

   if (iage>selpar_afull_HB){
                             sel_HB_1(iage)=1.0-(1.0-selpar_min_HB)/
                               (1.+mfexp(-1.*selpar_slope2_HB*(double(agebins(iage))-selpar_L502_HB)));
                             sel_HB_2(iage)=1.0-(1.0-selpar_min_HB)/
                               (1.+mfexp(-1.*selpar_slope2_HB*(double(agebins(iage))-selpar_L502_HB)));
                             sel_HB_3(iage)=1.0-(1.0-selpar_min_HB)/
                               (1.+mfexp(-1.*selpar_slope2_HB*(double(agebins(iage))-selpar_L502_HB)));
                            }

   if (iage<selpar_afull_PVT) {sel_PVT_2(iage)=1./(1.+mfexp(-1.*selpar_slope_PVT2*(double(agebins(iage))-selpar_L50_PVT2)));
                                sel_PVT_3(iage)=1./(1.+mfexp(-1.*selpar_slope_PVT3*(double(agebins(iage))-selpar_L50_PVT3)));
                               }
   if (iage>selpar_afull_PVT){sel_PVT_2(iage)=1.0-(1.0-selpar_min_PVT)/
                               (1.+mfexp(-1.*selpar_slope2_PVT*(double(agebins(iage))-selpar_L502_PVT)));
                               sel_PVT_3(iage)=1.0-(1.0-selpar_min_PVT)/
                               (1.+mfexp(-1.*selpar_slope2_PVT*(double(agebins(iage))-selpar_L502_PVT)));
                               }

  } //end age loop

 sel_initial=set_sel_initial_wgt_cL*sel_cL_2+L_hb_bias*set_sel_initial_wgt_HB*sel_HB_1+L_pvt_bias*set_sel_initial_wgt_PVT*sel_PVT_2;
 sel_initial=sel_initial/max(sel_initial);

//  //sel_cL_1=sel_cL_1/max(sel_cL_1);    //normalize to max of one (typically only needed for double logistic)
//  sel_cL_2=sel_cL_2/max(sel_cL_2);    //normalize to max of one
//  sel_cL_3=sel_cL_3/max(sel_cL_3);    //normalize to max of one
//
////  sel_cD_2=sel_cD_2/max(sel_cD_2);    //normalize to max of one
////  sel_cD_3=sel_cD_3/max(sel_cD_3);    //normalize to max of one
//
//  sel_HB_1=sel_HB_1/max(sel_HB_1);    //normalize to max of one
//  sel_HB_2=sel_HB_2/max(sel_HB_2);    //normalize to max of one
//  sel_HB_3=sel_HB_3/max(sel_HB_3);    //normalize to max of one
//
//  //sel_PVT_1=sel_HB_1;
//  sel_PVT_2=sel_PVT_2/max(sel_PVT_2); //normalize to max of one
//  sel_PVT_3=sel_PVT_3/max(sel_PVT_3); //normalize to max of one

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

  //Period 1:
  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
```

```
     sel_cL(iyear)=sel_cL_2; //early commercial handline sel same as in subsequent period
     sel_HB(iyear)=sel_HB_1;
     sel_PVT(iyear)=sel_PVT_2; //PVT same period 2.
  }

  //Period 2:
  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  {
     sel_cL(iyear)=sel_cL_2;
     sel_cD(iyear)=sel_cD_3; //sane as in period 3
     sel_HB(iyear)=sel_HB_2;
     sel_PVT(iyear)=sel_PVT_2;
  }

  //Period 3
  for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
  {
     sel_cL(iyear)=sel_cL_3;
     sel_cD(iyear)=sel_cD_3;
     sel_HB(iyear)=sel_HB_3;
     sel_PVT(iyear)=sel_PVT_3;
  }
//---Discard selectivities---------------------------------
//---------------------------------------------------------

//  for (iage=1; iage<=nages; iage++)
//  {
//  sel_HB_D_3(iage)=1./(1.+mfexp(-1.*selpar_slope_HB_D3*(double(agebins(iage))-selpar_L50_HB_D3))); //logistic
//  sel_HB_D_3(iage)=(1./(1.+mfexp(-1.*selpar_slope_HB_D3*(double(agebins(iage))-
//                  selpar_L50_HB_D3))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_HB_D3*
//                  (double(agebins(iage))-(selpar_L50_HB_D3+selpar_L502_HB_D3)))))); //double logistic
//  }
//  sel_HB_D_3=sel_HB_D_3/max(sel_HB_D_3); //re-normalize double logistic

  selpar_Age1_HB_D3=1.0/(1.0+mfexp(-selpar_Age1_HB_D3_logit));
  selpar_Age1_cL_D3=1.0/(1.0+mfexp(-selpar_Age1_cL_D3_logit));
  selpar_Age2_cL_D3=1.0/(1.0+mfexp(-selpar_Age2_cL_D3_logit));

  sel_HB_D_2(1)=selpar_Age1_HB_D3; //Assume same sel of age 1's across periods
  sel_HB_D_2(2)=1.0;
  sel_HB_D_3(1)=selpar_Age1_HB_D3;
  sel_HB_D_3(2)=1.0;
  for (iage=3; iage<=nages; iage++)
      {sel_HB_D_2(iage)=vecprob_HB_D2(iage);
       sel_HB_D_3(iage)=vecprob_HB_D3(iage);}

  sel_cL_D_3(1)=selpar_Age1_cL_D3;
  sel_cL_D_3(2)=selpar_Age2_cL_D3;
  sel_cL_D_3(3)=1.0;
  for (iage=4; iage<=nages; iage++)
      {sel_cL_D_3(iage)=vecprob_cL_D3(iage);}


//  //Period 1: No discards assumed in period 1
//  for (iyear=styr; iyear<=endyr_period1; iyear++)
//  {sel_HB_D(iyear)=sel_HB_D_2;}

  //Period 2: No commercial discards in period 2
  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  {     sel_HB_D(iyear)=sel_HB_D_2;}

  //Period 3: all fleets same as HB
  for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
  {sel_HB_D(iyear)=sel_HB_D_3;
   sel_cL_D(iyear)=sel_cL_D_3;}

  //PVC discard selectivity same as headboat;
  sel_PVT_D=sel_HB_D;

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  //initialization F is avg from first 2 yrs of observed landings (3 yr average typically preferable, but not so here)
  log_F_dev_init_cL=sum(log_F_dev_cL(styr_cL_L,(styr_cL_L+1)))/2.0;
  log_F_init_HB=sum(log_F_dev_HB(styr_HB_L,(styr_HB_L+1)))/2.0;
  log_F_dev_init_PVT=sum(log_F_dev_PVT(styr_PVT_L,(styr_PVT_L+1)))/2.0;

  //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_cD_L & iyear<=endyr_cD_L)
    {  F_cD_out(iyear)=mfexp(log_avg_F_cD+log_F_dev_cD(iyear)); //}
    // if (iyear<styr_cD_L) {F_cD_out(iyear)=0.0;}
       F_cD(iyear)=sel_cD(iyear)*F_cD_out(iyear);
       Fsum(iyear)+=F_cD_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_PVT_L & iyear<=endyr_PVT_L)
    {  F_PVT_out(iyear)=mfexp(log_avg_F_PVT+log_F_dev_PVT(iyear)); //}
    // if (iyear<styr_PVT_L){F_PVT_out(iyear)=mfexp(log_avg_F_PVT+log_F_dev_init_PVT);}
       F_PVT(iyear)=sel_PVT(iyear)*F_PVT_out(iyear);
       Fsum(iyear)+=F_PVT_out(iyear);
    }

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

    if(iyear>=styr_cL_D & iyear<=endyr_cL_D)
      {F_cL_D_out(iyear)=mfexp(log_avg_F_cL_D+log_F_dev_cL_D(iyear));}
    F_cL_D(iyear)=sel_cL_D(iyear)*F_cL_D_out(iyear);
    Fsum(iyear)+=F_cL_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_PVT_D& iyear<=endyr_PVT_D)
    { F_PVT_D_out(iyear)=mfexp(log_avg_F_PVT_D+log_F_dev_PVT_D(iyear));
      F_PVT_D(iyear)=sel_PVT_D(iyear)*F_PVT_D_out(iyear);
      Fsum(iyear)+=F_PVT_D_out(iyear);
    }

    //Total F at age
    F(iyear)=F_cL(iyear); //first in additive series (NO +=)
    F(iyear)+=F_cD(iyear);
    F(iyear)+=F_HB(iyear);
    F(iyear)+=F_PVT(iyear);

    F(iyear)+=F_cL_D(iyear);
    F(iyear)+=F_HB_D(iyear);
    F(iyear)+=F_PVT_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);
  if (set_BiasCor <= 0.0) {BiasCor=mfexp(rec_sigma_sq/2.0);}   //bias correction
  else {BiasCor=set_BiasCor;}


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


  F_initial=sel_initial*F_init*F_init_ratio;

//  F_initial=F_init_ratio*set_F_init;

  Z_initial=M+F_initial;


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

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

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

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



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

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

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

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

  SSB(styr)=sum(elem_prod(N_spawn(styr),reprod));
  MatFemB(styr)=sum(elem_prod(N_spawn(styr),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
    {
        //add dzero to avoid log(zero)
        //N(iyear+1,1)=BiasCor*mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
        //    (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero));
        N(iyear+1,1)=mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
            (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero)+log_rec_historic_dev(iyear+1));
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
        MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));

    }
    else   //recruitment follows S-R curve with lognormal deviation
    {
        //add dzero to avoid log(zero)
        N(iyear+1,1)=mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0*
            (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero)+log_rec_dev(iyear+1));
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
        MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
    }
  }


    //last year (projection) has no recruitment variability
    N(endyr+1,1)=mfexp(log(((0.8*R0*steep*SSB(endyr))/(0.2*R0*spr_F0*
                (1.0-steep)+(steep-0.2)*SSB(endyr)))+dzero));
    N(endyr+1)(2,nages)=++elem_prod(N(endyr)(1,nages-1),(mfexp(-1.*Z(endyr)(1,nages-1))));
    N(endyr+1,nages)+=N(endyr,nages)*mfexp(-1.*Z(endyr,nages));//plus group
    //SSB(endyr+1)=sum(elem_prod(N(endyr+1),reprod));


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

FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_cL_num(iyear,iage)=N(iyear,iage)*F_cL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cD_num(iyear,iage)=N(iyear,iage)*F_cD(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_PVT_num(iyear,iage)=N(iyear,iage)*F_PVT(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }

    pred_cL_L_knum(iyear)=sum(L_cL_num(iyear))/1000.0;
    pred_cD_L_knum(iyear)=sum(L_cD_num(iyear))/1000.0;
    pred_HB_L_knum(iyear)=sum(L_HB_num(iyear))/1000.0;
    pred_PVT_L_knum(iyear)=sum(L_PVT_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_cD_klb(iyear)=elem_prod(L_cD_num(iyear),wgt_cD_klb(iyear));     //in 1000 lb
    L_HB_klb(iyear)=elem_prod(L_HB_num(iyear),wgt_HB_klb(iyear));     //in 1000 lb
    L_PVT_klb(iyear)=elem_prod(L_PVT_num(iyear),wgt_PVT_klb(iyear));  //in 1000 lb

    pred_cL_L_klb(iyear)=sum(L_cL_klb(iyear));
    pred_cD_L_klb(iyear)=sum(L_cD_klb(iyear));
    pred_HB_L_klb(iyear)=sum(L_HB_klb(iyear));
    pred_PVT_L_klb(iyear)=sum(L_PVT_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_cL_num(iyear,iage)=N(iyear,iage)*F_cL_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_PVT_num(iyear,iage)=N(iyear,iage)*F_PVT_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_cL_D_knum(iyear)=sum(D_cL_num(iyear))/1000.0;             //pred annual dead discards in 1000s (for matching data)
    pred_cL_D_klb(iyear)=sum(elem_prod(D_cL_num(iyear),wgt_cL_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_PVT_D_knum(iyear)=sum(D_PVT_num(iyear))/1000.0;           //pred annual dead discards in 1000s (for matching data)
    pred_PVT_D_klb(iyear)=sum(elem_prod(D_PVT_num(iyear),wgt_PVT_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------------------------


 //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 number units
      N_HB(iyear)=elem_prod(N_mdyr(iyear),sel_HB(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

 //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 discards
  for (iyear=1;iyear<=nyr_cL_D_lenc;iyear++) //all yrs within period 3
  {pred_cL_D_lenc(iyear)=(D_cL_num(yrs_cL_D_lenc(iyear))*lenprob_cL_D3)
                        /sum(D_cL_num(yrs_cL_D_lenc(iyear)));
  }
```

```
  //Commercial dv
  for (iyear=1;iyear<=nyr_cD_lenc;iyear++) //all yrs within period 3
  {pred_cD_lenc(iyear)=(L_cD_num(yrs_cD_lenc(iyear))*lenprob_cD3)
                        /sum(L_cD_num(yrs_cD_lenc(iyear)));
  }


 //Headboat (for-hire)
  for (iyear=1;iyear<=nyr_HB_lenc;iyear++)
  {  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
  {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)));
  }


 //Compute weighted pooled length comps for PVT
 L_PVT_num_pool.initialize();
 for (iyear=1;iyear<=nyr_PVT_lenc_pool;iyear++)
 {L_PVT_num_pool_yr(iyear)=nsamp_PVT_lenc_pool(iyear)*L_PVT_num(yrs_PVT_lenc_pool(iyear))
                           /sum(L_PVT_num(yrs_PVT_lenc_pool(iyear)));
  if (yrs_PVT_lenc_pool(iyear)<=endyr_period2) {L_PVT_num_pool(1)+=L_PVT_num_pool_yr(iyear);}
  if (yrs_PVT_lenc_pool(iyear)>endyr_period2) {L_PVT_num_pool(2)+=L_PVT_num_pool_yr(iyear);}
 }
 //PVT All in periods 2,3
  for (iyear=1;iyear<=nyr_PVT_lenc;iyear++) //all yrs within periods 2,3
  {  if (yrs_PVT_lenc(iyear)<=endyr_period2)
      {pred_PVT_lenc(iyear)=(L_PVT_num_pool(iyear)*lenprob_PVT2)/sum(L_PVT_num_pool(iyear));}
     if (yrs_PVT_lenc(iyear)>endyr_period2)
      {pred_PVT_lenc(iyear)=(L_PVT_num_pool(iyear)*lenprob_PVT3)/sum(L_PVT_num_pool(iyear));}
  }

FUNCTION get_age_comps


 //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 dive
  for (iyear=1;iyear<=nyr_cD_agec;iyear++)
  {
    ErrorFree_cD_agec(iyear)=L_cD_num(yrs_cD_agec(iyear))/
                          sum(L_cD_num(yrs_cD_agec(iyear)));
    pred_cD_agec(iyear)=age_error*ErrorFree_cD_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);
  }

 //PVT
  for (iyear=1;iyear<=nyr_PVT_agec;iyear++)
  {
    ErrorFree_PVT_agec(iyear)=L_PVT_num(yrs_PVT_agec(iyear))/
                        sum(L_PVT_num(yrs_PVT_agec(iyear)));
    pred_PVT_agec(iyear)=age_error*ErrorFree_PVT_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_cD+
        sum(log_F_dev_cD((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_PVT+
        sum(log_F_dev_PVT((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cL_D+
        sum(log_F_dev_cL_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_PVT_D+
        sum(log_F_dev_PVT_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_cD_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cD+
        sum(log_F_dev_cD((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_PVT_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_PVT+
        sum(log_F_dev_PVT((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cL_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cL_D+
        sum(log_F_dev_cL_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_PVT_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_PVT_D+
        sum(log_F_dev_PVT_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_cD=sum(log_F_dev_cD((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_PVT=sum(log_F_dev_PVT((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

  log_F_dev_end_cL_D=sum(log_F_dev_cL_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_PVT_D=sum(log_F_dev_PVT_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_cD(endyr)*mfexp(log_avg_F_cD+log_F_dev_end_cD)+
          sel_HB(endyr)*mfexp(log_avg_F_HB+log_F_dev_end_HB)+
          sel_PVT(endyr)*mfexp(log_avg_F_PVT+log_F_dev_end_PVT);

  F_end_D=sel_cL_D(endyr)*mfexp(log_avg_F_cL_D+log_F_dev_end_cL_D)+
          sel_HB_D(endyr)*mfexp(log_avg_F_HB_D+log_F_dev_end_HB_D)+
          sel_PVT_D(endyr)*mfexp(log_avg_F_PVT_D+log_F_dev_end_PVT_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_cD_prop+F_HB_prop+F_PVT_prop;
  wgt_wgted_L_klb=F_cL_prop/wgt_wgted_L_denom*wgt_cL_klb(endyr)+
              F_cD_prop/wgt_wgted_L_denom*wgt_cD_klb(endyr)+
              F_HB_prop/wgt_wgted_L_denom*wgt_HB_klb(endyr)+
              F_PVT_prop/wgt_wgted_L_denom*wgt_PVT_klb(endyr);

  wgt_wgted_D_denom=F_cL_D_prop+F_HB_D_prop+F_PVT_D_prop;
  wgt_wgted_D_klb=F_cL_D_prop/wgt_wgted_D_denom*wgt_cL_D_klb(endyr)+
              F_HB_D_prop/wgt_wgted_D_denom*wgt_HB_D_klb(endyr)+
              F_PVT_D_prop/wgt_wgted_D_denom*wgt_PVT_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_mdyr(1,(nages-1))=elem_prod(N_age_msy(1,(nages-1)),
                                   mfexp((-1.*Z_age_msy(1,(nages-1)))*spawn_time_frac));
    N_age_msy_mdyr(nages)=(N_age_msy_mdyr(nages-1)*
                          (mfexp(-1.*(Z_age_msy(nages-1)*(1.0-spawn_time_frac) +
                                 Z_age_msy(nages)*spawn_time_frac) )))
                          /(1.0-mfexp(-1.*Z_age_msy(nages)));

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


    //Compute equilibrium values of R (including bias correction), SSB and Yield at each F
    R_eq(ff)=(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_mdyr*=R_eq(ff);

    for (iage=1; iage<=nages; iage++)
    {
      L_age_msy(iage)=N_age_msy(iage)*(F_L_age_msy(iage)/Z_age_msy(iage))*
                      (1.-mfexp(-1.*Z_age_msy(iage)));
      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_mdyr,reprod));
    B_eq(ff)=sum(elem_prod(N_age_msy,wgt_mt));
    L_eq_klb(ff)=sum(elem_prod(L_age_msy,wgt_wgted_L_klb));
    L_eq_knum(ff)=sum(L_age_msy)/1000.0;
    D_eq_klb(ff)=sum(elem_prod(D_age_msy,wgt_wgted_D_klb));
    D_eq_knum(ff)=sum(D_age_msy)/1000.0;
  }

  msy_klb_out=max(L_eq_klb);

  for(ff=1; ff<=n_iter_msy; ff++)
```

```
  {
   if(L_eq_klb(ff) == msy_klb_out)
      {
        SSB_msy_out=SSB_eq(ff);
        B_msy_out=B_eq(ff);
        R_msy_out=R_eq(ff);
        msy_knum_out=L_eq_knum(ff);
        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); //pow(var_rec_dev,0.5);  //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_cD_L_klb(iyear)+
                               pred_HB_L_klb(iyear)+pred_PVT_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_cL_L_knum(iyear)+pred_cD_L_knum(iyear)+
                               pred_HB_L_knum(iyear)+pred_PVT_L_knum(iyear);

        D_total_knum_yr(iyear)+=pred_cL_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_cL_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_PVT_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_PVT_D_klb(iyear);

        D_cL_klb(iyear)=elem_prod(D_cL_num(iyear),wgt_cL_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_PVT_klb(iyear)=elem_prod(D_PVT_num(iyear),wgt_PVT_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_HB_num+L_PVT_num+L_cL_num+L_cD_num); //landings at age in number fish
  L_total_klb=L_HB_klb+L_PVT_klb+L_cL_klb+L_cD_klb;   //landings at age in klb whole weight

  D_total_num=(D_HB_num+D_PVT_num+D_cL_num);          //discards at age in number fish
  D_total_klb=D_HB_klb+D_PVT_klb+D_cL_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)*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);}
   for(iyear=(styr+1); iyear<=(styr_rec_dev-1); iyear++)
     {log_rec_historic_dev_output(iyear)=log_rec_historic_dev(iyear);}
   //fill in log Nage deviations for ages they are nonzero (ages2+)
   for(iage=2; iage<=nages; iage++)
   {
    log_Nage_dev_output(iage)=log_Nage_dev(iage);
   }


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

  //static per-recruit stuff

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

```
    N_age_spr(nages)=N_age_spr(nages)/(1.0-mfexp(-1.*Z(iyear,nages)));
    N_age_spr_mdyr(1,(nages-1))=elem_prod(N_age_spr(1,(nages-1)),
                                mfexp(-1.*Z(iyear)(1,(nages-1))*spawn_time_frac));
    N_age_spr_mdyr(nages)=(N_age_spr_mdyr(nages-1)*
                          (mfexp(-1.*(Z(iyear)(nages-1)*(1.0-spawn_time_frac) + Z(iyear)(nages)*spawn_time_frac) )))
                          /(1.0-mfexp(-1.*Z(iyear)(nages)));
    spr_static(iyear)=sum(elem_prod(N_age_spr_mdyr,reprod))/spr_F0;
  }


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

    F_L_age_spr=F_spr(ff)*sel_wgted_L;

    Z_age_spr=M+F_L_age_spr+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_mdyr(1,(nages-1))=elem_prod(N_age_spr(1,(nages-1)),
                                   mfexp((-1.*Z_age_spr(1,(nages-1)))*spawn_time_frac));
    N_age_spr_mdyr(nages)=(N_age_spr_mdyr(nages-1)*
                          (mfexp(-1.*(Z_age_spr(nages-1)*(1.0-spawn_time_frac) + Z_age_spr(nages)*spawn_time_frac) )))
                          /(1.0-mfexp(-1.*Z_age_spr(nages)));

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

FUNCTION get_effective_sample_sizes

      neff_cL_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
      neff_cL_D_lenc_allyr_out=missing;
      neff_cD_lenc_allyr_out=missing;
      neff_HB_lenc_allyr_out=missing;
      neff_HB_D_lenc_allyr_out=missing;
      neff_PVT_lenc_allyr_out=missing;
      neff_cL_agec_allyr_out=missing;
      neff_cD_agec_allyr_out=missing;
      neff_HB_agec_allyr_out=missing;
      neff_PVT_agec_allyr_out=missing;

      for (iyear=1; iyear<=nyr_cL_lenc; iyear++)
         {if (nsamp_cL_lenc(iyear)>=minSS_cL_lenc)
            { numer=sum( elem_prod(pred_cL_lenc(iyear),(1.0-pred_cL_lenc(iyear))) );
              denom=sum( square(obs_cL_lenc(iyear)-pred_cL_lenc(iyear)) );
                if (denom>0.0) {neff_cL_lenc_allyr_out(yrs_cL_lenc(iyear))=numer/denom;}
                else {neff_cL_lenc_allyr_out(yrs_cL_lenc(iyear))=-missing;}
            } else {neff_cL_lenc_allyr_out(yrs_cL_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cL_D_lenc; iyear++)
         {if (nsamp_cL_D_lenc(iyear)>=minSS_cL_D_lenc)
            { numer=sum( elem_prod(pred_cL_D_lenc(iyear),(1.0-pred_cL_D_lenc(iyear))) );
              denom=sum( square(obs_cL_D_lenc(iyear)-pred_cL_D_lenc(iyear)) );
                if (denom>0.0) {neff_cL_D_lenc_allyr_out(yrs_cL_D_lenc(iyear))=numer/denom;}
                else {neff_cL_D_lenc_allyr_out(yrs_cL_D_lenc(iyear))=-missing;}
            } else {neff_cL_D_lenc_allyr_out(yrs_cL_D_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cD_lenc; iyear++)
         {if (nsamp_cD_lenc(iyear)>=minSS_cD_lenc)
            { numer=sum( elem_prod(pred_cD_lenc(iyear),(1.0-pred_cD_lenc(iyear))) );
              denom=sum( square(obs_cD_lenc(iyear)-pred_cD_lenc(iyear)) );
                if (denom>0.0) {neff_cD_lenc_allyr_out(yrs_cD_lenc(iyear))=numer/denom;}
                else {neff_cD_lenc_allyr_out(yrs_cD_lenc(iyear))=-missing;}
            } else {neff_cD_lenc_allyr_out(yrs_cD_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
         {if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
            { numer=sum( elem_prod(pred_HB_lenc(iyear),(1.0-pred_HB_lenc(iyear))) );
              denom=sum( square(obs_HB_lenc(iyear)-pred_HB_lenc(iyear)) );
                if (denom>0.0) {neff_HB_lenc_allyr_out(yrs_HB_lenc(iyear))=numer/denom;}
                else {neff_HB_lenc_allyr_out(yrs_HB_lenc(iyear))=-missing;}
            } 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_HB_D_lenc)
            { numer=sum( elem_prod(pred_HB_D_lenc(iyear),(1.0-pred_HB_D_lenc(iyear))) );
              denom=sum( square(obs_HB_D_lenc(iyear)-pred_HB_D_lenc(iyear)) );
                if (denom>0.0) {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=numer/denom;}
                else {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=-missing;}
            } else {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_PVT_lenc; iyear++)
         {if (nsamp_PVT_lenc(iyear)>=minSS_PVT_lenc)
            { numer=sum( elem_prod(pred_PVT_lenc(iyear),(1.0-pred_PVT_lenc(iyear))) );
```

```
              denom=sum( square(obs_PVT_lenc(iyear)-pred_PVT_lenc(iyear)) );
                if (denom>0.0) {neff_PVT_lenc_allyr_out(yrs_PVT_lenc(iyear))=numer/denom;}
                else {neff_PVT_lenc_allyr_out(yrs_PVT_lenc(iyear))=-missing;}
            } else {neff_PVT_lenc_allyr_out(yrs_PVT_lenc(iyear))=-99;}
         }


      for (iyear=1; iyear<=nyr_cL_agec; iyear++)
         {if (nsamp_cL_agec(iyear)>=minSS_cL_agec)
            { numer=sum( elem_prod(pred_cL_agec(iyear),(1.0-pred_cL_agec(iyear))) );
              denom=sum( square(obs_cL_agec(iyear)-pred_cL_agec(iyear)) );
                if (denom>0.0) {neff_cL_agec_allyr_out(yrs_cL_agec(iyear))=numer/denom;}
                else {neff_cL_agec_allyr_out(yrs_cL_agec(iyear))=-missing;}
            } else {neff_cL_agec_allyr_out(yrs_cL_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_cD_agec; iyear++)
         {if (nsamp_cD_agec(iyear)>=minSS_cD_agec)
            { numer=sum( elem_prod(pred_cD_agec(iyear),(1.0-pred_cD_agec(iyear))) );
              denom=sum( square(obs_cD_agec(iyear)-pred_cD_agec(iyear)) );
                if (denom>0.0) {neff_cD_agec_allyr_out(yrs_cD_agec(iyear))=numer/denom;}
                else {neff_cD_agec_allyr_out(yrs_cD_agec(iyear))=-missing;}
            } else {neff_cD_agec_allyr_out(yrs_cD_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_HB_agec; iyear++)
         {if (nsamp_HB_agec(iyear)>=minSS_HB_agec)
            { numer=sum( elem_prod(pred_HB_agec(iyear),(1.0-pred_HB_agec(iyear))) );
              denom=sum( square(obs_HB_agec(iyear)-pred_HB_agec(iyear)) );
                if (denom>0.0) {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=numer/denom;}
                else {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=-missing;}
            } else {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_PVT_agec; iyear++)
         {if (nsamp_PVT_agec(iyear)>=minSS_PVT_agec)
            { numer=sum( elem_prod(pred_PVT_agec(iyear),(1.0-pred_PVT_agec(iyear))) );
              denom=sum( square(obs_PVT_agec(iyear)-pred_PVT_agec(iyear)) );
                if (denom>0.0) {neff_PVT_agec_allyr_out(yrs_PVT_agec(iyear))=numer/denom;}
                else {neff_PVT_agec_allyr_out(yrs_PVT_agec(iyear))=-missing;}
            } else {neff_PVT_agec_allyr_out(yrs_PVT_agec(iyear))=-99;}
         }



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


FUNCTION evaluate_objective_function
  fval=0.0;
  fval_data=0.0;
//fval=square(xdum-9.0); //used in model development


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

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

  f_HBD_cpue=0.0;
  for (iyear=styr_HBD_cpue; iyear<=endyr_HBD_cpue; iyear++)
  {
    f_HBD_cpue+=square(log((pred_HBD_cpue(iyear)+dzero)/
        (obs_HBD_cpue(iyear)+dzero)))/(2.0*log(1.0+square(HBD_cpue_cv(iyear)/w_I_HBD)));
  }
  fval+=f_HBD_cpue;
  fval_data+=f_HBD_cpue;

  f_cL_cpue=0.0;
  for (iyear=styr_cL_cpue; iyear<=endyr_cL_cpue; iyear++)
  {
    f_cL_cpue+=square(log((pred_cL_cpue(iyear)+dzero)/
        (obs_cL_cpue(iyear)+dzero)))/(2.0*log(1.0+square(cL_cpue_cv(iyear)/w_I_cL)));
  }
  fval+=f_cL_cpue;
  fval_data+=f_cL_cpue;

  f_HB_cpue=0.0;
  for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
  {
    f_HB_cpue+=square(log((pred_HB_cpue(iyear)+dzero)/
        (obs_HB_cpue(iyear)+dzero)))/(2.0*log(1.0+square(HB_cpue_cv(iyear)/w_I_HB)));
  }
  fval+=f_HB_cpue;
  fval_data+=f_HB_cpue;


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

  f_cL_L=0.0; //in 1000 lb ww
  for (iyear=styr_cL_L; iyear<=endyr_cL_L; iyear++)
  {
      f_cL_L+=square(log((pred_cL_L_klb(iyear)+dzero)/
        (obs_cL_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*cL_L_cv(iyear)/w_L)));
  }
  fval+=f_cL_L;
  fval_data+=f_cL_L;

  f_cD_L=0.0; //in 1000 lb ww
  for (iyear=styr_cD_L; iyear<=endyr_cD_L; iyear++)
  {
    f_cD_L+=square(log((pred_cD_L_klb(iyear)+dzero)/
        (obs_cD_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*cD_L_cv(iyear)/w_L)));
  }
```

```
  fval+=f_cD_L;
  fval_data+=f_cD_L;

  f_HB_L=0.0; //in 1000 fish
  for (iyear=styr_HB_L; iyear<=1980; iyear++)
  {
    f_HB_L+=square(log((pred_HB_L_knum(iyear)+dzero)/
        (L_hb_bias*obs_HB_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*HB_L_cv(iyear)/w_L)));
  }
  for (iyear=1981; iyear<=endyr_HB_L; iyear++)
  {
    f_HB_L+=square(log((pred_HB_L_knum(iyear)+dzero)/
        (obs_HB_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*HB_L_cv(iyear)/w_L)));
  }
  fval+=f_HB_L;
  fval_data+=f_HB_L;

  f_PVT_L=0.0; //in 1000 fish
  for (iyear=styr_PVT_L; iyear<=1980; iyear++)
  {
    f_PVT_L+=square(log((pred_PVT_L_knum(iyear)+dzero)/
        (L_pvt_bias*obs_PVT_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*PVT_L_cv(iyear)/w_L)));
  }
  for (iyear=1981; iyear<=endyr_PVT_L; iyear++)
  {
    f_PVT_L+=square(log((pred_PVT_L_knum(iyear)+dzero)/
        (obs_PVT_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*PVT_L_cv(iyear)/w_L)));
  }
  fval+=f_PVT_L;
  fval_data+=f_PVT_L;


//---Discards-------------------------------
  f_cL_D=0.0; //in 1000 fish
  for (iyear=styr_cL_D; iyear<=endyr_cL_D; iyear++)
  {
    f_cL_D+=square(log((pred_cL_D_knum(iyear)+dzero)/
        (obs_cL_D(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*cL_D_cv(iyear)/w_D)));
  }
  fval+=f_cL_D;
  fval_data+=f_cL_D;


  f_HB_D=0.0; //in 1000 fish
  for (iyear=styr_HB_D; iyear<=endyr_HB_D; iyear++)
  {
    f_HB_D+=square(log((pred_HB_D_knum(iyear)+dzero)/
        (obs_HB_D(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*HB_D_cv(iyear)/w_D)));
  }
  fval+=f_HB_D;
  fval_data+=f_HB_D;

  f_PVT_D=0.0; //in 1000 fish
  for (iyear=styr_PVT_D; iyear<=endyr_PVT_D; iyear++)
  {
    f_PVT_D+=square(log((pred_PVT_D_knum(iyear)+dzero)/
        (obs_PVT_D(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*PVT_D_cv(iyear)/w_D)));
  }
  fval+=f_PVT_D;
  fval_data+=f_PVT_D;


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

  f_cL_lenc=0.;
  for (iyear=1; iyear<=nyr_cL_lenc; iyear++)
  {
    if (nsamp_cL_lenc(iyear)>=minSS_cL_lenc)
    {
    f_cL_lenc-=w_lc_cL*neff_cL_lenc(iyear)*
         sum(elem_prod((obs_cL_lenc(iyear)+dzero),
             log(elem_div((pred_cL_lenc(iyear)+dzero),
                (obs_cL_lenc(iyear)+dzero)))));
    }
  }
  fval+=f_cL_lenc;
  fval_data+=f_cL_lenc;


  f_cL_D_lenc=0.;
  for (iyear=1; iyear<=nyr_cL_D_lenc; iyear++)
  {
    if (nsamp_cL_D_lenc(iyear)>=minSS_cL_D_lenc)
    {
    f_cL_D_lenc-=w_lc_cL_D*neff_cL_D_lenc(iyear)*
         sum(elem_prod((obs_cL_D_lenc(iyear)+dzero),
             log(elem_div((pred_cL_D_lenc(iyear)+dzero),
                (obs_cL_D_lenc(iyear)+dzero)))));
    }
  }
  fval+=f_cL_D_lenc;
  fval_data+=f_cL_D_lenc;


  f_cD_lenc=0.;
  for (iyear=1; iyear<=nyr_cD_lenc; iyear++)
  {
    if (nsamp_cD_lenc(iyear)>=minSS_cD_lenc)
    {
    f_cD_lenc-=w_lc_cD*neff_cD_lenc(iyear)*
         sum(elem_prod((obs_cD_lenc(iyear)+dzero),
             log(elem_div((pred_cD_lenc(iyear)+dzero),
                (obs_cD_lenc(iyear)+dzero)))));
    }
```

```
  }
  fval+=f_cD_lenc;
  fval_data+=f_cD_lenc;


  f_HB_lenc=0.;
  for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
  {
    if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
    {
    f_HB_lenc-=w_lc_HB*neff_HB_lenc(iyear)*
         sum(elem_prod((obs_HB_lenc(iyear)+dzero),
             log(elem_div((pred_HB_lenc(iyear)+dzero),
                (obs_HB_lenc(iyear)+dzero)))));
    }
  }
  fval+=f_HB_lenc;
  fval_data+=f_HB_lenc;


  f_HB_D_lenc=0.;
  for (iyear=1; iyear<=nyr_HB_D_lenc; iyear++)
  {
    if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
    {
    f_HB_D_lenc-=w_lc_HB_D*neff_HB_D_lenc(iyear)*
         sum(elem_prod((obs_HB_D_lenc(iyear)+dzero),
             log(elem_div((pred_HB_D_lenc(iyear)+dzero),
                (obs_HB_D_lenc(iyear)+dzero)))));
    }
  }
  fval+=f_HB_D_lenc;
  fval_data+=f_HB_D_lenc;


  f_PVT_lenc=0.;
  for (iyear=1; iyear<=nyr_PVT_lenc; iyear++)
  {
    if (nsamp_PVT_lenc(iyear)>=minSS_PVT_lenc)
    {
    f_PVT_lenc-=w_lc_PVT*neff_PVT_lenc(iyear)*
         sum(elem_prod((obs_PVT_lenc(iyear)+dzero),
             log(elem_div((pred_PVT_lenc(iyear)+dzero),
                (obs_PVT_lenc(iyear)+dzero)))));
    }
  }
  fval+=f_PVT_lenc;
  fval_data+=f_PVT_lenc;


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

  f_cL_agec=0.0;
  for (iyear=1; iyear<=nyr_cL_agec; iyear++)
  {
    if (nsamp_cL_agec(iyear)>=minSS_cL_agec)
    {
    f_cL_agec-=w_ac_cL*neff_cL_agec(iyear)*
            sum(elem_prod((obs_cL_agec(iyear)+dzero),
               log(elem_div((pred_cL_agec(iyear)+dzero),
                  (obs_cL_agec(iyear)+dzero)))));
    }
  }
  fval+=f_cL_agec;
  fval_data+=f_cL_agec;

  f_cD_agec=0.0;
  for (iyear=1; iyear<=nyr_cD_agec; iyear++)
  {
    if (nsamp_cD_agec(iyear)>=minSS_cD_agec)
    {
    f_cD_agec-=w_ac_cD*neff_cD_agec(iyear)*
            sum(elem_prod((obs_cD_agec(iyear)+dzero),
               log(elem_div((pred_cD_agec(iyear)+dzero),
                  (obs_cD_agec(iyear)+dzero)))));
    }
  }
  fval+=f_cD_agec;
  fval_data+=f_cD_agec;

  f_HB_agec=0.0;
  for (iyear=1; iyear<=nyr_HB_agec; iyear++)
  {
    if (nsamp_HB_agec(iyear)>=minSS_HB_agec)
    {
    f_HB_agec-=w_ac_HB*neff_HB_agec(iyear)*
            sum(elem_prod((obs_HB_agec(iyear)+dzero),
                log(elem_div((pred_HB_agec(iyear)+dzero),
                   (obs_HB_agec(iyear)+dzero)))));
    }
  }
  fval+=f_HB_agec;
  fval_data+=f_HB_agec;


  f_PVT_agec=0.0;
  for (iyear=1; iyear<=nyr_PVT_agec; iyear++)
  {
    if (nsamp_PVT_agec(iyear)>=minSS_PVT_agec)
    {
    f_PVT_agec-=w_ac_PVT*neff_PVT_agec(iyear)*
            sum(elem_prod((obs_PVT_agec(iyear)+dzero),
                log(elem_div((pred_PVT_agec(iyear)+dzero),
                   (obs_PVT_agec(iyear)+dzero)))));
```

```
    }
  }
  fval+=f_PVT_agec;
  fval_data+=f_PVT_agec;


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

  f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
  if (w_rec_early>0.0)
    { if (styr_rec_dev<endyr_rec_phase1)
        {
          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_sq/2.0)/
                            (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_sq/2.0)/
                            (2.0*rec_sigma_sq)) + rec_logL_add;}
        }
      fval+=w_rec_end*f_rec_dev_end;
   }

  f_rec_historic_dev=norm2(log_rec_historic_dev);
  fval+=f_rec_historic_dev;


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


//  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 (!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;


  //Random walk components of fishery dependent indices
  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---------------------------------------------------
//If no se available, CV=1 (ie, se=mu) assumed (loose prior)
  f_priors=0.0;
  f_priors+=square(len_sd_val-set_len_sd)/square(set_len_sd_se);
  f_priors+=square(steep-set_steep)/square(set_steep_se);
  f_priors+=square(rec_sigma-set_rec_sigma)/square(set_rec_sigma_se);
  f_priors+=square(R_autocorr-set_R_autocorr);
  f_priors+=square(q_DD_beta-set_q_DD_beta)/square(set_q_DD_beta_se);
//  f_priors+=square(q_rate-set_q_rate)/(dzero+square(set_q_rate));
  f_priors+=square(F_init-set_F_init)/square(set_F_init);
//  f_priors+=square(M_constant-set_M_constant)/square(set_M_constant_se);


  f_priors+=square(selpar_L50_cL2-set_selpar_L50_cL2)/square(set_selpar_L50_cL2);
  f_priors+=square(selpar_slope_cL2-set_selpar_slope_cL2)/square(set_selpar_slope_cL2);
  f_priors+=square(selpar_L50_cL3-set_selpar_L50_cL3)/square(set_selpar_L50_cL3);
  f_priors+=square(selpar_slope_cL3-set_selpar_slope_cL3)/square(set_selpar_slope_cL3);
  f_priors+=square(selpar_L502_cL-set_selpar_L502_cL)/square(set_selpar_L502_cL);
  f_priors+=square(selpar_slope2_cL-set_selpar_slope2_cL)/square(set_selpar_slope2_cL);
  f_priors+=square(selpar_min_cL-set_selpar_min_cL)/(dzero+square(set_selpar_min_cL));

  f_priors+=square(selpar_Age1_cL_D3_logit-prior_selpar_Age1_cL_D3_logit);
  f_priors+=square(selpar_Age2_cL_D3_logit-prior_selpar_Age2_cL_D3_logit);

  f_priors+=square(selpar_L50_cD3-set_selpar_L50_cD3)/square(set_selpar_L50_cD3);
  f_priors+=square(selpar_slope_cD3-set_selpar_slope_cD3)/square(set_selpar_slope_cD3);
  f_priors+=square(selpar_L502_cD-set_selpar_L502_cD)/square(set_selpar_L502_cD);
  f_priors+=square(selpar_slope2_cD-set_selpar_slope2_cD)/square(set_selpar_slope2_cD);
  f_priors+=square(selpar_min_cD-set_selpar_min_cD)/(dzero+square(set_selpar_min_cD));


  f_priors+=square(selpar_L50_HB1-set_selpar_L50_HB1)/square(set_selpar_L50_HB1);
  f_priors+=square(selpar_slope_HB1-set_selpar_slope_HB1)/square(set_selpar_slope_HB1);
  f_priors+=square(selpar_L50_HB2-set_selpar_L50_HB2)/square(set_selpar_L50_HB2);
  f_priors+=square(selpar_slope_HB2-set_selpar_slope_HB2)/square(set_selpar_slope_HB2);
  f_priors+=square(selpar_L50_HB3-set_selpar_L50_HB3)/square(set_selpar_L50_HB3);
  f_priors+=square(selpar_slope_HB3-set_selpar_slope_HB3)/square(set_selpar_slope_HB3);
  f_priors+=square(selpar_L502_HB-set_selpar_L502_HB)/square(set_selpar_L502_HB);
  f_priors+=square(selpar_slope2_HB-set_selpar_slope2_HB)/square(set_selpar_slope2_HB);
  f_priors+=square(selpar_min_HB-set_selpar_min_HB)/(dzero+square(set_selpar_min_HB));

  f_priors+=square(selpar_Age1_HB_D3_logit-prior_selpar_Age1_HB_D3_logit);

  f_priors+=square(selpar_L50_PVT2-set_selpar_L50_PVT2)/square(set_selpar_L50_PVT2);
  f_priors+=square(selpar_slope_PVT2-set_selpar_slope_PVT2)/square(set_selpar_slope_PVT2);
  f_priors+=square(selpar_L50_PVT3-set_selpar_L50_PVT3)/square(set_selpar_L50_PVT3);
  f_priors+=square(selpar_slope_PVT3-set_selpar_slope_PVT3)/square(set_selpar_slope_PVT3);
  f_priors+=square(selpar_L502_PVT-set_selpar_L502_PVT)/square(set_selpar_L502_PVT);
  f_priors+=square(selpar_slope2_PVT-set_selpar_slope2_PVT)/square(set_selpar_slope2_PVT);
  f_priors+=square(selpar_min_PVT-set_selpar_min_PVT)/(dzero+square(set_selpar_min_PVT));

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


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 << "F" << endl;
    //  report << F <<endl;

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

  } //endl last phase loop
```

Appendix B AD Model Builder input file for the Beaufort Assessment Model

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##  Data Input File
##  SEDAR24 Assessment: Red Snapper
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#starting and ending year of model
1955
2009
#Starting year to estimate recruitment deviation from S-R curve
1975
#3 phases of constraints on recruitment deviations:
#allows possible heavier constraint (weights defined later) in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1977
2008
#3 blocks of size regs: yr1-82 no restrictions, 1983(midyr)-91 12-inch TL, 1992-09 20-in TL
#ending years of regulation blocks
1982
1991
#Size limits of blocks 2, 3 (in mm: 1mm=0.0394in)
304.5685 #block 2
507.6142 #block 3
304.5685 #release size applied to discards in block 2, typically would be set to size limit in block 2

#Number of ages (20 classes is 1,...,20+)
20
#vector of agebins, last is a plus group
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

#number length bins used to match length comps and number used to compute plus group
28
10

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
190 220 250 280 310 340 370 400 430 460 490 520 550 580 610 640 670 700 730 760 790 820 850 880 910 940 970 1000
1030 1060 1090 1120 1150 1180 1210 1240 1270 1300


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

#####################################################################################################
###Headboat discard index -- at-sea observer  (HBD)#####################################################################
#Starting and ending years of time series, respectively
2005
2009
#Observed CPUE (numbers) and CV vectors, respectively
0.56 0.41 2.02 1.39 0.63
0.30 0.37 0.17 0.21 0.27

#####################################################################################################
#######################Commercial Hook and Line fishery landings#########################
#Commercial Hook and Line CPUE Index from Logbook
#Starting and ending years of CPUE index
1993
2009
#Observed CPUE and assumed CVs
1.14 0.91 0.92 0.57 0.57 0.63 0.76 0.75 1.22 1.37 1.11 1.44 1.23 0.61 0.66 1.20 1.92
0.06 0.05 0.05 0.06 0.06 0.06 0.06 0.06 0.05 0.05 0.05 0.05 0.06 0.07 0.07 0.07 0.07
#Commercial Hook and Line fishery landings
#Starting and ending years of landings time series, respectively
1955
2009
#Observed landings (1000 lb whole weight) and assumed CVs
497.800 484.300 868.900 617.300 662.700 677.100 799.800 662.577 504.840 559.491 656.795 740.057 963.706 1069.332 700.493 640.918 543.433 468.602 387.344
632.507 745.363 619.011 649.273 589.918 409.939 380.596 371.379 306.128 310.268 248.195 240.971 215.743 187.211 164.123 258.478 215.047 134.032 89.062
189.994 179.615 166.772 130.650 101.232 80.009 80.506 92.109 175.233 163.092 118.803 149.791 118.015 80.291 104.737 240.735 341.241
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
###Starting and ending years of discards time series, respectively
1992
2009
###Observed discards (1000 fish) and assumed CVs
14.233 14.926 20.638 19.437 24.867 27.458 21.106 19.387 18.975 19.014 42.356 13.973 5.17 4.999 7.425 14.759 15.512 20.402
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
###Starting and ending years of commercial hook and line length composition sample data
#1985 #1986 starts TIP in FL
#Number and vector of years of length compositions for hook and line fishery
13
1985 1986 1989 1990 1991 1992 1993 1994 1995 1997 2001 2002 2003
#sample size of commercial length comp data by year    (first row observed N, second row effective N: effective may be set to observed)
153.0 90.0 105.0 98.0 149.0 89.0 128.0 132.0 145.0 84.0 168.0 167.0 223.0
153.0 90.0 105.0 98.0 149.0 89.0 128.0 132.0 145.0 84.0 168.0 167.0 223.0
#commercial length composition samples (year,lengthbin 3 cm)
```

```
0.000000 0.000000 0.001017 0.000290 0.005618 0.026770 0.060149 0.094637 0.178793 0.186368 0.122924 0.096595 0.057901
0.030044 0.022845 0.012858 0.007132 0.004218 0.002917 0.008042 0.006525 0.012841 0.015958 0.016744 0.022379 0.006435
0.000000 0.000000
0.000000 0.000000 0.000000 0.002075 0.003377 0.032727 0.022332 0.025885 0.095691 0.173115 0.173366 0.114787 0.072566
0.058285 0.059346 0.049562 0.034851 0.014790 0.006945 0.002025 0.009126 0.008083 0.008311 0.017943 0.009546 0.004023
0.001245 0.000000
0.000000 0.000000 0.000843 0.000421 0.000421 0.007767 0.013084 0.042319 0.075495 0.100157 0.158119 0.173811 0.161640
0.095636 0.038433 0.032777 0.011754 0.011975 0.014260 0.016332 0.012759 0.010644 0.009618 0.005633 0.002529 0.003573
0.000000 0.000000
0.000000 0.000000 0.002593 0.028526 0.010405 0.041394 0.139546 0.130049 0.108801 0.101634 0.099010 0.069709 0.042585
0.040140 0.031024 0.021239 0.023011 0.008876 0.013156 0.016834 0.005111 0.023714 0.018686 0.009208 0.011816 0.002934
0.000000 0.000000
0.000000 0.000000 0.000000 0.001855 0.024024 0.149096 0.168815 0.104315 0.097915 0.059435 0.068701 0.078838 0.036045
0.027792 0.023545 0.027794 0.020494 0.013408 0.009447 0.010389 0.006727 0.013764 0.025280 0.013731 0.014961 0.003629
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001080 0.006035 0.021349 0.083301 0.186991 0.100973
0.091720 0.104090 0.071163 0.065079 0.031521 0.029083 0.048929 0.038168 0.036941 0.027857 0.024832 0.006153 0.024736
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000681 0.001363 0.000000 0.001277 0.000595 0.004209 0.138139 0.348584 0.207642
0.106410 0.048978 0.022516 0.019412 0.023805 0.012039 0.008577 0.007123 0.018205 0.009255 0.008071 0.007639 0.002740
0.002739 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001958 0.080060 0.191697 0.180820
0.178203 0.158656 0.101014 0.043367 0.009721 0.007753 0.004327 0.008588 0.006129 0.007197 0.006826 0.009139 0.004546
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000493 0.000000 0.091985 0.235512 0.131339
0.132387 0.088036 0.076163 0.062600 0.056821 0.017551 0.020354 0.007948 0.010843 0.034580 0.011236 0.013291 0.006648
0.002212 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.065525 0.068298 0.021787
0.153009 0.101968 0.108294 0.093697 0.158273 0.060929 0.065351 0.050050 0.019140 0.008186 0.005464 0.020027 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.008080 0.038195 0.037460 0.019097 0.024239 0.010283 0.015552 0.136224 0.191938 0.133316
0.110714 0.088498 0.064647 0.040347 0.018284 0.009829 0.011796 0.013967 0.009409 0.006895 0.006692 0.001405 0.003133
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000988 0.004942 0.084648 0.223367 0.171110
0.143702 0.122290 0.068797 0.082527 0.042138 0.017463 0.013904 0.009604 0.010776 0.001817 0.000963 0.000963 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001288 0.000000 0.001881 0.004151 0.033698 0.103314 0.098755
0.160390 0.160672 0.138549 0.103635 0.060823 0.052718 0.029870 0.017218 0.007625 0.010785 0.005454 0.003727 0.002872
0.002575 0.000000
###Number and vector of years of age compositions for hook and line fishery
8
1996 1997 1998 1999 2000 2004 2007 2009
###sample sizes of age comps by year  (first row observed N, second row effective N: effective may be set to observed)
58.0 144.0 37.0 156.0 257.0 30.0 138.0 294.0
58.0 144.0 37.0 156.0 257.0 30.0 138.0 294.0
#age composition samples (year,age)
0.000000 0.014380 0.145825 0.078870 0.187001 0.247596 0.203717 0.029253 0.029885 0.008108 0.006911 0.001441 0.008653 0.013186 0.001441 0.000000 0.004558 0.000000 0.000000 0.019175
0.000000 0.015905 0.032149 0.339226 0.216568 0.167788 0.127107 0.039095 0.018722 0.003212 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.040227
0.000000 0.109016 0.340646 0.119042 0.203438 0.072064 0.037353 0.011826 0.053577 0.017668 0.016473 0.001224 0.000000 0.000000 0.003806 0.008836 0.000000 0.000000 0.000000 0.005031
0.000000 0.012889 0.492218 0.321494 0.079304 0.021659 0.020658 0.018861 0.005936 0.012023 0.001847 0.003060 0.000000 0.002215 0.000000 0.000000 0.000000 0.000000 0.000000 0.007836
0.000000 0.011195 0.350711 0.431766 0.068031 0.040021 0.030049 0.030672 0.006715 0.019137 0.002835 0.000000 0.000000 0.000000 0.002835 0.000000 0.000000 0.000000 0.000759 0.005272
0.000000 0.077347 0.350106 0.403970 0.088105 0.077934 0.000924 0.000000 0.001614 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.295774 0.056044 0.052911 0.145506 0.091871 0.121759 0.123058 0.063609 0.024290 0.000000 0.001933 0.000000 0.004131 0.008077 0.001910 0.001910 0.000000 0.000000 0.007216
0.000000 0.019517 0.292256 0.375970 0.008835 0.012204 0.062204 0.046928 0.049304 0.048592 0.030063 0.017376 0.004155 0.005699 0.001507 0.003996 0.011777 0.004121 0.000000 0.005495
#Number and vector of years of length compositions for commercial hook and line discards
1
2007 #input as 2007
#sample size of commercial discard length comp data by year    (first row observed N, second row effective N: effective may be set to observed)
6.0
6.0
#commercial discard length composition samples (year,lengthbin 3 cm)
0.000000 0.000000 0.000000 0.000000 0.007752 0.007752 0.085271 0.224806 0.240310 0.224806 0.178295 0.023256 0.000000 0.000000
0.000000 0.000000 0.007752 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000


#####################################################################################################
#################################Commercial Diving fishery landings ########### ########### ########### ###########
#Starting and ending years of landings time series, respectively
1984
2009
#Observed landings (1000 lb whole weight)  and CV's
1.317 2.547 0.508 0.030 0.013 0.006 1.859 5.898 9.614 5.611 13.116 10.037 6.153 7.531 8.063 9.974 10.376 18.238 22.097 17.454 19.647
9.344 4.163 7.514 6.304 8.011
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05
#Number and vector of years of length compositions (comm dv)
3
1999    2000   2003
###sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
13.0 9.0 12.0
13.0 9.0 12.0
#commercial dive length comp samples (year,age) (3cm length bins)
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.086701 0.086490 0.136317
0.123665 0.037123 0.098587 0.098648 0.061657 0.024627 0.012356 0.012275 0.061477 0.049240 0.073893 0.024616 0.012330
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.011600 0.207320 0.196233 0.172812
0.080528 0.068605 0.045736 0.034277 0.045780 0.034277 0.000000 0.011408 0.045702 0.034288 0.011434 0.000000 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.005107 0.005129 0.091415 0.126964 0.116753
0.106638 0.106915 0.070928 0.096469 0.086020 0.071133 0.045511 0.020292 0.035537 0.005070 0.005070 0.005049 0.000000
0.000000 0.000000
#Number and vector of years of age compositions (diving)
3
2000 2001 2009
#sample sizes of age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
124.0 30.0 17.0
124.0 30.0 17.0
#diving age comp samples (year,age)
0.000000 0.118083 0.385875 0.314938 0.134779 0.000000 0.015424 0.000000 0.030900 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.580221 0.171249 0.231437 0.017093 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.515661 0.394191 0.000000 0.000000 0.000000 0.000000 0.045074 0.000000 0.000000 0.015025 0.015025 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.015025


########################################For hire (headboat+charterboat) landings########################################
```

```
#Starting and ending years for CPUE index
1976
2009
#Observed CPUE values (numbers) and CVs,
2.30 2.24 2.11 2.12 1.42 2.88 1.14 1.53 1.31 1.99 0.47 0.56 0.54 0.91 0.84 0.65 0.08 0.15 0.26 0.28 0.25 0.27 0.24 0.30 0.42 0.80 0.96 0.53 0.83 0.80 0.45 0.46 1.86 2.04
0.07 0.07 0.05 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.06 0.05 0.05 0.06 0.07 0.07 0.07 0.06 0.07 0.08 0.06 0.06 0.06 0.06 0.06 0.07 0.05 0.06 0.06 0.06 0.05 0.05
###Starting and ending years for landings time series
1955
2009
#For-hire landings vector (1000 fish) and CV’s
68.301 74.807 81.321 84.472 85.598 85.480 83.527 79.441 76.530 78.771 86.525 96.861 104.809 104.716 95.537 82.889 71.743 65.493 65.872
71.612 77.286 78.829 75.868 68.640 58.535 47.760 69.519 37.726 59.229 60.094 97.119 98.995 40.286 62.664 44.461 26.656 30.623 45.611
14.948 22.589 22.423 8.681 62.935 18.112 49.363 19.508 21.879 30.115 23.899 24.796 23.113 17.293 17.326 41.780 50.210
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Starting and ending years of discards time series, respectively
1983
2009
#Observed discards (1000s) and assumed CVs
42.281 121.668 27.775 0.158 0.158 0.158 0.158 0.158 0.697 17.936 33.397 7.359 24.366 5.053 19.038 8.856 47.594 32.530 32.845 25.886 21.700 37.465 49.435 23.194 118.249 59.846 35.131
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Number and vector of years of length compositions for for-hire
14
1976 1977 1988 1990 1991 1992 1993 1994 1995 1997 1998 1999 2000 2001
#sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
115.0 195.0 128.0 140.0 71.0 55.0 107.0 83.0 84.0 54.0 92.0 113.0 94.0 151.0
115.0 195.0 128.0 140.0 71.0 55.0 107.0 83.0 84.0 54.0 92.0 113.0 94.0 151.0
#for-hire length comp samples (year,lengthbin)
0.000000 0.006441 0.041867 0.077294 0.061191 0.109499 0.161029 0.157808 0.106399 0.061430 0.055518 0.056309 0.025924
0.011578 0.003139 0.006721 0.008744 0.015112 0.003532 0.003340 0.004588 0.004660 0.007304 0.007112 0.000000 0.003221
0.000240 0.000000
0.000000 0.022379 0.043036 0.037872 0.056808 0.086072 0.115337 0.142880 0.170573 0.106826 0.051751 0.027887 0.017462
0.009650 0.020098 0.011414 0.016148 0.016159 0.014190 0.005304 0.012147 0.001829 0.006972 0.007111 0.000097 0.000000
0.000000 0.000000
0.000000 0.010610 0.008018 0.017686 0.064180 0.127055 0.175536 0.200757 0.079377 0.052385 0.069698 0.013021 0.026309
0.025191 0.021651 0.006519 0.029717 0.021808 0.012110 0.024862 0.000000 0.000000 0.000000 0.008207 0.005305 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000572 0.077616 0.156988 0.175170 0.224627 0.103173 0.081302 0.040663 0.030294 0.019532
0.029632 0.030477 0.007067 0.001875 0.000000 0.003760 0.000000 0.000000 0.004873 0.010503 0.001875 0.000000 0.000000
0.000000 0.000000
0.001265 0.000000 0.016077 0.048155 0.048943 0.177689 0.106122 0.131047 0.148125 0.096097 0.082089 0.013533 0.045638
0.028535 0.011983 0.001265 0.001779 0.011317 0.000000 0.001965 0.003145 0.007247 0.000000 0.006788 0.011195 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.008031 0.000000 0.000000 0.008779 0.004341 0.501067 0.028643 0.161926 0.104971 0.013291
0.041861 0.024977 0.030519 0.008068 0.038587 0.000000 0.000000 0.005787 0.005787 0.006683 0.006683 0.000000 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.004650 0.006982 0.000000 0.033184 0.153096 0.304697 0.210393
0.065655 0.049603 0.064482 0.022946 0.010836 0.020251 0.018650 0.009291 0.006549 0.000000 0.003877 0.013913 0.000000
0.000947 0.000000
0.000000 0.000000 0.000000 0.000000 0.000262 0.000262 0.000000 0.000000 0.000000 0.006402 0.042918 0.213224 0.240037
0.176481 0.054688 0.069593 0.094932 0.007918 0.032753 0.029894 0.010847 0.000475 0.003411 0.000000 0.005217 0.010688
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.002309 0.004617 0.004980 0.000000 0.016904 0.004871 0.051994 0.172239 0.220794
0.210199 0.091574 0.063670 0.046415 0.058394 0.022362 0.001003 0.004871 0.000502 0.011882 0.005210 0.000000 0.005210
0.000000 0.000000
0.000000 0.000000 0.000000 0.033443 0.115630 0.022288 0.001386 0.000000 0.000000 0.019516 0.072692 0.064000 0.055662
0.059865 0.149004 0.071868 0.057208 0.030740 0.188678 0.001592 0.019493 0.008567 0.017672 0.010697 0.000000 0.000000
0.000000 0.000000
0.000000 0.000000 0.000116 0.000116 0.011045 0.005025 0.005451 0.000233 0.000710 0.002454 0.083177 0.309851 0.287135
0.130650 0.058530 0.000000 0.024259 0.013195 0.020578 0.013195 0.006278 0.004553 0.004922 0.004224 0.000000 0.000000
0.000000 0.014300
0.000000 0.000000 0.000000 0.001836 0.031884 0.048546 0.026033 0.009484 0.011089 0.009498 0.091618 0.202689 0.157678
0.177273 0.099031 0.033931 0.024866 0.039302 0.014535 0.005055 0.007536 0.001040 0.007074 0.000000 0.000000 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000636 0.002535 0.006903 0.002715 0.003144 0.017371 0.142350 0.223327 0.234330
0.157116 0.141236 0.037850 0.009613 0.015235 0.001596 0.000000 0.001767 0.000000 0.001478 0.000000 0.000798 0.000000
0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.003405 0.000000 0.003405 0.010373 0.047419 0.230923 0.213266
0.223592 0.103071 0.067247 0.027629 0.027328 0.010965 0.005514 0.014239 0.000387 0.000032 0.000274 0.010700 0.000231
0.000000 0.000000
#Number and vector of years of age compositions (for-hire)
20
1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1989 1996 2002 2003 2004 2005 2006 2007 2008 2009
#sample sizes of age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
83.0 32.0 36.0 145.0 56.0 173.0 178.0 161.0 100.0 64.0 32.0 58.0 105.0 108.0 98.0 130.0 123.0 51.0 52.0 359.0
83.0 32.0 36.0 145.0 56.0 173.0 178.0 161.0 100.0 64.0 32.0 58.0 105.0 108.0 98.0 130.0 123.0 51.0 52.0 359.0
#for-hire age comp samples (year,age)
0.017240 0.516094 0.410147 0.029586 0.018886 0.004399 0.002931 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000716
0.000000 0.778527 0.107437 0.036894 0.033540 0.036894 0.006708 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.124946 0.621428 0.175569 0.065265 0.000000 0.012495 0.000000 0.000297 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.038314 0.731378 0.197550 0.013540 0.006275 0.007669 0.000000 0.002201 0.000682 0.000000 0.000000 0.000000 0.000682 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001708
0.051173 0.489784 0.306549 0.134680 0.008293 0.001060 0.000000 0.004780 0.003541 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000140 0.000000 0.000000
0.326862 0.568206 0.094220 0.004430 0.002499 0.001872 0.000852 0.000776 0.000146 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000006 0.000131
0.143123 0.703762 0.136527 0.004464 0.005136 0.000356 0.000881 0.000000 0.000352 0.000352 0.000884 0.000801 0.001059 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.002302
0.070643 0.745726 0.173827 0.007643 0.000369 0.000535 0.000000 0.000230 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000230 0.000000 0.000460 0.000085 0.000251
0.088675 0.482581 0.366815 0.056387 0.001928 0.001313 0.001071 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001230
0.138813 0.277068 0.522050 0.050347 0.011722 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.069538 0.683527 0.228635 0.003577 0.000902 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000647 0.000000 0.000000 0.013174 0.000000 0.000000 0.000000
0.000000 0.000000 0.236095 0.181631 0.155831 0.202269 0.138661 0.038004 0.000000 0.020314 0.023416 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.003779
0.000000 0.273611 0.559616 0.132688 0.030590 0.001279 0.000792 0.000062 0.000134 0.000000 0.000040 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.001188
0.000000 0.127347 0.507910 0.272120 0.036590 0.022254 0.001150 0.024540 0.000000 0.007777 0.000312 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.080166 0.665299 0.187411 0.049246 0.009169 0.004552 0.001934 0.000281 0.000000 0.000114 0.000000 0.000000 0.001809 0.000000 0.000000 0.000000 0.000000 0.000000 0.000020
0.000000 0.018179 0.558290 0.307046 0.075633 0.026488 0.008538 0.005382 0.000000 0.000201 0.000000 0.000243 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.023488 0.368882 0.490387 0.081730 0.016468 0.004963 0.001137 0.002407 0.000603 0.000010 0.000000 0.005542 0.000010 0.000000 0.000000 0.000165 0.001803 0.000000 0.002406
0.000000 0.445552 0.199636 0.298734 0.039638 0.000920 0.000382 0.014228 0.000000 0.000909 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.119981 0.850865 0.020326 0.000584 0.006745 0.000914 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000584 0.000000 0.000000 0.000000
0.000000 0.063902 0.620991 0.297642 0.003687 0.001777 0.004108 0.001642 0.001151 0.001066 0.000919 0.001189 0.000765 0.000383 0.000000 0.000176 0.000000 0.000259 0.000124 0.000222
#Number and vector of years of headboat discard length composition data
5
2005 2006 2007 2008 2009
#sample sizes of length comps by year (first row observed N, second row effective N: effective may be set to observed)
44.0 30.0 65.0 63.0 56.0
44.0 30.0 65.0 63.0 56.0
```

```
#HB discard length composition by year (year,lengthbin 3cm)
0.000000 0.002041 0.018367 0.044898 0.030612 0.128571 0.157143 0.167347 0.173469 0.200000 0.067347 0.008163 0.000000
0.002041 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000
0.001488 0.007440 0.046131 0.117560 0.220238 0.238095 0.215774 0.099702 0.025298 0.019345 0.005952 0.001488 0.000000
0.001488 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000
0.001335 0.016689 0.022697 0.056075 0.098131 0.127503 0.229640 0.231642 0.139519 0.062083 0.012684 0.001335 0.000000
0.000668 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000
0.002982 0.005963 0.020274 0.048301 0.150865 0.156231 0.157424 0.156828 0.136553 0.125224 0.035778 0.002385 0.000596
0.000000 0.000596 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000
0.000000 0.002304 0.023041 0.036866 0.057604 0.131336 0.152074 0.131336 0.195853 0.202765 0.062212 0.004608 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000

#####################################################################################################
######################################PVT recreational landings  #################################
#Private recreational landings
#Starting and ending years for landings time series
1955
2009
#PVT landings vector (1000 fish) and CVs
13.763 18.067 22.657 26.582 30.115 33.277 35.672 37.195 39.544 44.904 53.626 64.051 72.901 76.108 72.701 66.731 62.080 61.735 67.536
78.477 89.063 94.852 95.145 89.822 80.445 69.978 121.730 52.932 43.885 161.385 178.659 78.195 51.281 98.608 107.354 11.091 31.351 38.345
10.864 13.567 2.386 11.419 3.545 7.585 22.660 57.664 40.185 33.865 16.111 25.390 21.172 14.541 31.324 84.502 92.814
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Starting and ending years of discards time series, respectively
1983
2009
#Observed discards (1000s) and assumed CVs
8.679 22.845 63.501 8.679 106.560 48.373 20.038 8.679 35.853 19.492 48.989 62.577 37.932 17.628 8.679 22.970 132.663 223.334 179.264 105.891 139.401 163.953 79.725 115.593 339.128 352.213 183.886
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Number and vector of years of PVT length composition data (these are pooled data, 1985=1983-1991; 2000=1992-2009
2
1985  2000
#sample sizes of length comp data by year  (first row observed N, second row effective N: effective may be set to observed)
79.0 165.0
79.0 165.0
#PVT length comp samples (year,lengthbin)
0.000000 0.000000 0.037906 0.093932 0.269840 0.084935 0.121840 0.101896 0.053529 0.012650 0.055284 0.029981
0.059865 0.030637 0.023628 0.001173 0.004753 0.003788 0.000000 0.000000 0.000000 0.000628 0.013108 0.000000
0.000000 0.000628 0.000000 0.000000
0.000000 0.000000 0.000000 0.001669 0.001742 0.011922 0.012453 0.015399 0.011868 0.017867 0.014277 0.038231
0.171215 0.200243 0.163867 0.104824 0.085102 0.040463 0.034814 0.011311 0.027496 0.005979 0.006538 0.006852
0.004836 0.004542 0.001557 0.004933
#Number and vector of years of PVT length composition data used to weight annual predictions before pooling
26
1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009
#sample sizes of length comp data by year
3.0 9.0 14.0 9.0 10.0 16.0 12.0 4.0 2.0 5.0 4.0 2.0 2.0 3.0 5.0 11.0 13.0 14.0 9.0 6.0 12.0 7.0 8.0 8.0 31.0 25.0
#Starting and ending year of PVT age composition data
1
2009
#sample sizes of PVT age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
11.0
11.0
###PVT age comps (year,lengthbin)
0.000000 0.000000 0.140365 0.794530 0.061389 0.000000 0.003717 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000


######################Biological input ###############################################################################
##discard mortality constant
0.48  #comm HAL
0.41  #headboat
0.39  #PVT

#VonBert params (Linf, K, t0), units in mm TL
902.0
0.24
-0.03
#Standard errors of vonBert param (Linf, K, t0), applied if params are estimated
4.29
0.004
0.03
#sd of length at age
51.0
#standard error of SD of length at age, applied if sd is estimated
51.0

#length-weight (TL-whole wgt) coefficients a and b, W=aL^b, (W in g, TL in mm)
7.15E-06
3.12
#weight-gonad weight (whole wgt-gondad weight) coefficients a and b, GW=aW^b (units=g)
3.1416E-05
1.743
#time-invariant vector of % maturity-at-age for females (ages 1-20)
0.221 0.549 0.839 0.957 0.990 0.998 0.999 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
#time-invariant vector of proportion female (ages 1-20)
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
#time of year (as fraction) for spawning: end of July/beginning of August=7/12
0.583
#age-dependent natural mortality at age
0.30 0.17 0.13 0.11 0.10 0.09 0.09 0.08 0.08 0.08 0.08 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
#age-independent natural mortality (used only to compute MSST=(1-M)SSBmsy)
0.08
#SE of age-independent natural mortality (=(0.08-0.0.05)/1.96 ... sd chosen to put closest bound at 95%CI)
0.015
#Max observed age
54
#Spawner-recruit parameters
```

```
#steepness (fixed or initial guess) (0.75 from meta-analysis)
0.85
#SE of steepness (from meta-analysis)
0.15
#log_R0 - log virgin recruitment
13.0
# R autocorrelation
0.0
# SD of recruitment in log space
0.6
# SE of SD recruitment
0.3


##################Parameter values and initial guesses##############################################################################
###Selectivity parameters.
###Initial guess must be within boundaries.
# Initial guesses initialized near solutions from preliminary model runs
# age at size limits (12, 20 inches)= 1.66, 3.36
# zero in slope2 provides logistic selectivity
# Not all selectivity parameters are used in the base-run model

2.0 #selpar_L50_cL2
1.5 #selpar_slope_cL2
3.5 #selpar_L50_cL3
2.0 #selpar_slope_cL3
8.0 #selpar_L502_cL3
1.0 #selpar_slope2_cL3
0.5 #selpar_min_cL3
5   #selpar_afull_cL3  an integer value

0.05 #selpar_Age1_cL_D3
0.5 #selpar_Age2_cL_D3

2.0 #selpar_L50_cD2
1.5 #selpar_slope_cD2
3.0 #selpar_L50_cD3
2.0 #selpar_slope_cD3
8.0 #selpar_L502_cD3
1.0 #selpar_slope2_cD3
0.5 #selpar_min_cD3
4   #selpar_afull_cD3  an integer value

1.5 #selpar_L50_HB1
3.0 #selpar_slope_HB1
1.5 #selpar_L50_HB2
3.0 #selpar_slope_HB2
3.1 #selpar_L50_HB3
2.0 #selpar_slope_HB3
8.0 #selpar_L502_HB3
1.5 #selpar_slope2_HB3
0.3 #selpar_min_HB3
3   #selpar_afull_HB3 (an integer value)

0.5 #selpar_Age1_HB_D3

1.5 #selpar_L50_PVT2
3.0 #selpar_slope_PVT2
3.1 #selpar_L50_PVT3
2.0 #selpar_slope_PVT3
8.0 #selpar_L502_PVT3
1.5 #selpar_slope2_PVT3 (not used: descending limb mirrors for-hire fleet)
0.3 #selpar_min_PVT3 (not used: descending limb mirrors for-hire fleet)
3   #selpar_afull_PVT3  an integer value (not used: descending limb mirrors for-hire fleet)

55.5  #weight of comm to initial sel (mean landings in knum (1953-1955)
63.0  #weight of for-hire to initial sel (mean landings in knum (1953-1955)
10.3  #weight of private to initial sel (mean landings in knum (1953-1955)


###################Likelihood Component Weighting##################################################################################
##Weights in objective fcn
1.0  #landings
1.0  #discards
0.49  #cL length comps
0.49  #cL discard length comps
0.79     #cD length comps
0.09  #HB length comps
0.10  #HB discard length comps
0.09  #PVT length comps
0.07  #cL age comps
1.20  #cD age comps
0.01  #HB age comps
0.01  #PVT age comps
0.60  #HBD index
0.16   #cL index
0.11  #HB index
1.0     #S-R residuals
0.0  #constraint on early recruitment deviations
0.0     #constraint on ending recruitment deviations
0.0     #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization)
0.0     #weight on tuning F (penalty not applied in final phase of optimization)
#0.0  #penalty on deviation in CV at age
#0.0     #penalty on first difference in CV at age

#bias adjustment (multiplier) for historic for-hire and private rec landings, in that order
1.0
1.0

#log catchabilities (initial guesses)
-13.0     #HBD
-8.0      #commHAL (index in weight)
-13.0     #HB
#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
```

```
-1
##annual positive rate of increase on all fishery dependent q's due to technology creep
0.02
# DD q switch: Integer value (choose estimation phase, negative value turns it off)
-1
##density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)
0.0
##SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)
0.128
#Age to begin counting D-D q (should be age near full exploitation)
4
#Random walk switch:Integer value (choose estimation phase, negative value turns it off)
-3
#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence
0.03
0.03
0.03

##log mean F's (initial guesses)
-3.0   #commHAL
-4.5   #comm DV
-3.0   #HB
-2.0   #PVT


#Initial F (Input here, could also use mean of first few years)
0.02
#Initialization F as a proportion of F from either input initial F or computed (e.g., first few assessment years)
1.0


#log mean F's for discards (initial guesses)
-4.0   #commHAL discards
-4.0   #HB discards
-2.5   #PVT discards

#multiplicative adjustment to expand or conctract CVs on landings/discards
1.0

#Tuning F (not applied in last phase of optimization)
0.8
#Year for tuning F
2006


##threshold sample sizes for length comps (set to 99999.0 if sel is fixed)
1.0 #c HAL
1.0 #c HAL discards
1.0 #c DV
1.0 #HB
1.0 #HB discards
1.0 #PVT

#threshold sample sizes (greater than or equal to) for age comps
1.0 #HAL
1.0 #Dive
1.0 #for-hire
1.0 #PVT

#Ageing error matrix (columns are true age 1-20, rows are ages as read for age comps: columns should sum to one)
#0.85833647 0.14014000 0.00126230 0.00000117 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.14136553 0.71837100 0.15411000 0.00206470 0.00000313 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00029794 0.14014000 0.68925500 0.16599100 0.00306288 0.00000701 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000001 0.00067470 0.15411000 0.66388500 0.17606400 0.00422396 0.00001363 0.00000001 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000007 0.00126230 0.16599100 0.64173900 0.18459300 0.00550922 0.00002380 0.00000002 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000034 0.00206470 0.17606400 0.62235300 0.19181400 0.00687994 0.00003811 0.00000005 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000117 0.00306288 0.18459300 0.60532700 0.19793500 0.00830074 0.00005695 0.00000009 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000313 0.00422396 0.19181400 0.59032300 0.20313300 0.00974125 0.00008043 0.00000015 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000701 0.00550922 0.19793500 0.57705700 0.20755600 0.01117660 0.00010843 0.00000024 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00001363 0.00687994 0.20313300 0.56529100 0.21133100 0.01258730 0.00014063 0.00000037
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00002380 0.00830074 0.20755600 0.55482300 0.21456200 0.01395840 0.00017658
#0.00000053 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000001 0.00003811 0.00974125 0.21133100 0.54548400 0.21733500 0.01527920
#0.00021572 0.00000074 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000002 0.00005695 0.01117660 0.21456200 0.53713100 0.21972200
#0.01654230 0.00025742 0.00000098 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000005 0.00008043 0.01258730 0.21733500 0.52964300
#0.22178400 0.01774320 0.00030109 0.00000127 0.00000000 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000009 0.00010843 0.01395840 0.21972200
#0.52291400 0.22357000 0.01887910 0.00034611 0.00000160 0.00000000
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000015 0.00014063 0.01527920
#0.22178400 0.51685700 0.22512200 0.01994940 0.00039193 0.00000196
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000024 0.00017658
#0.01654230 0.22357000 0.51139300 0.22647500 0.02095420 0.00043804
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000037
#0.00021572 0.01774320 0.22512200 0.50645700 0.22765700 0.02922498
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000053 0.00025742 0.01887910 0.22647500 0.50199100 0.30525317
#0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#0.00000000 0.00000074 0.00030109 0.01994940 0.22765700 0.66464205

1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0

0.0 #p_lenc_cL2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_cL3 proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_cD2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_cD3  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_HB2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_HB3  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_PVT2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_PVT3  proportion of length comp mass below size limit considered when matching length comp

0.0 #p_lenc_cL_D3  proportion of length comp mass above size limit considered when matching length comp of discards
0.0 #p_lenc_HB_D2  proportion of length comp mass above size limit considered when matching length comp of discards
0.0 #p_lenc_HB_D3  proportion of length comp mass above size limit considered when matching length comp of discards
0.0 #p_lenc_PVT_D2  proportion of length comp mass above size limit considered when matching length comp of discards
0.0 #p_lenc_PVT_D3  proportion of length comp mass above size limit considered when matching length comp of discards

999 #end of data file flag
```