# SEDAR 19-RW-01: A statistical catch-age model for red grouper: mathematical description, implementation details, and computer code.

Sustainable Fisheries Branch

National Marine Fisheries Service

Southeast Fisheries Science Center

NOAA Center for Coastal Fisheries and Habitat Research

101 Pivers Island Road, Beaufort, NC 28516

## 1 Note

This document updates the model code submitted prior to the assessment workshop (working paper SEDAR19-AW-07) to reflect changes implemented during the AW.

## 2 Overview

The primary model in this assessment was a statistical catch-at-age model (Quinn and Deriso 1999), implemented with the AD Model Builder software (Otter Research 2004). In essence, a statistical catch-at-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-at-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-at-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 past 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, red snapper, vermilion snapper, and Spanish mackerel. The present version of this code, customiazed for SEDAR 19 red grouper, includes a number of improvements, including addition of a capability for modeling time-varying and/or density-dependent catchability, as suggested by a recent 2009 SEDAR procedural workshop on catchability.

## 3 Model configurations and equations

Model equations are detailed in Table 1, and AD Model Builder code for implementation is supplied in Appendix 1. An input data file for red grouper is included as Appendix 2. A general description of the assessment model follows:

**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 $\mathrm{W}_a$ by the power function $\mathrm{M}_a = \alpha W_a^{\beta}$, where $\alpha$ is a scale parameter and $\beta$ is a shape parameter. Lorenzen (1996) provided point estimates of $\alpha$ and $\beta$ 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 (26 years) as would occur with constant $M = 0.14$ from the DW. This approach using cumulative mortality is consistent with the findings of Hoenig (1983) and Hewitt and Hoenig (2005).

**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 $1 - 16^{+}$, where the oldest age class $16^{+}$ allowed for the accumulation of fish (i.e., plus group). Initial numbers at age were estimated in the model, but were penalized if they deviated from the stable age structure that resulted by assuming a constant, historical fishing mortality equal to the geometric mean fishing mortality for the first three years following model implementation (i.e., the geometric mean fishing mortality from 1976-1978).

**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 CV 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 in the population at large.

**Sex transition** Red grouper are a protogynous hermaphrodite. Transition from female to male was modeled with a logistic function, estimated by the DW.

**Maturity** Female maturity was modeled with a logistic function; parameters for this model were provided by the DW and treated as input to the assessment model. Males were assumed to be fully mature starting at age two (all age-1 fish were considered female).

**Spawning biomass** Spawning biomass was modeled as the weight of mature individuals in the population (males+females) at the time of spawning, where sex ratio at age was provided by the DW. For red grouper, peak spawning was considered to occur in mid-April.

**Recruitment** Recruitment was predicted from spawning biomass using a Beverton–Holt spawner-recruit model. Steepness, $h$, is a key parameter of this model; however, because it is often difficult to estimate steepness ($h$) reliably (Conn et al. In Review), a prior distribution was used to restrict estimates of $h$ to plausible values (SEDAR 2009b). Estimated recruitment was conditioned on the Beverton–Holt model, potentially with autocorrelated residuals.

**Landings** Time series of landing from four fisheries were modeled: commercial line (handline, longline), commercial other (pots, traps, trawl, diving, miscellaneous), headboat, and general recreational (MRFSS). 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 fisheries, and 1000 fish for headboats/recreational).

**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 0.2 for all fisheries, as suggested by the DW.

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

**Selectivities** In most cases, selectivities were estimated using a parametric approach. For most fisheries and indices, selectivities were estimated as a two-parameter logistic model. Exceptions were the commercial-other fishery and the MARMAP index, which were assumed to have dome-shaped (double logistic) selectivity, and required estimating four parameters. The commercial-other fishing mode partially mirrored the MARMAP selectivity (three of the four parameters were assumed equal). The inflection of the ascending limb was set equal to the age associated with a given regulatory period's size limit in this case. This parametric approach reduces the number of estimated parameters and imposes theoretical structure on the estimates. Critical to estimating selectivity parameters are age and size composition data. Prior distributions were used on selectivity slope parameters to stabilize estimation and keep them within reasonable values.

Selectivity of each fishery was fixed within each period of size-limit regulations, but was permitted to vary among periods. Fisheries experienced three periods of size-limit regulations (no limit prior to 1984, 12-inch limit during 1984–1991, and 20-inch limit from 1992-2008). Ideally, a model would have sufficient age composition data from each fishery over time to estimate 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 MRFSS collected little age or length composition data on red grouper until recently, headboat and general recreational fisheries were assumed to have the same selectivities in the first two regulatory periods. Commercial lines selectivity was also set equal for the first and second regulatory period.

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 discard selectivities was that the value for age-1 fish was estimated, age-2 fish were assumed to have full selection, and selectivity for older age fish was set at the age-specific probability of being below a fixed size (e.g., the size limit) given the estimated normal distribution of size at age. Given available data on discards, some additional assumptions were necessary; in particular, all fisheries were assumed to have the same discard selectivity function.

**Indices of abundance** The model was fit to two fishery independent indices of abundance (a Florida Keys RVC survey 1994–2008; and a MARMAP chevron trap survey 1991–2008) and to three fishery dependent indices of abundance (headboat 1978–2008; MRFSS 1991–2008; and commercial lines 1993–2008). 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 implemented for the red grouper assessment following recommendations of a 2009 SEDAR procedural workshop on catchability. In particular, capabilities for including density dependence, linear trends, and random walks were implemented. Parameters for these models could be estimated or fixed based on a priori considerations.

**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 ($F_{\mathrm{MSY}}$), and spawning biomass (total mature 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 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.

In addition to likelihoods, several penalties and prior distributions were included in the compound objective function. In some cases, as with stock-recruit autocorrelation, steepness, and selectivity slope parameters, priors were applied. Penalties on initial age-structure, catchability variation, and recruitment deviations were also implemented. 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.

We included a capability for each component of the likelihood to be weighted by user-supplied values (for instance, to give more influence to desired data sources). However, for initial runs of the red grouper assessment model, all weights were set to 1.0 with exception of $\omega_7$ and $\omega_8$, which were set to 0.0 (effectively removing penalties on initial and final recruitment deviations).

**Model testing** Experiments with a reduced model structure indicated that parameters estimated from the "Beaufort" model were unbiased and could be recovered from simulated data with little noise (cf., SEDAR 2007). Further, the general model structure has passed several rounds of independent peer review. As an additional measure of quality control, red grouper code and input data were examined by multiple analysts to ensure accuracy. This combination of testing and verification procedures suggest that the assessment model is implemented correctly and can provide an accurate assessment of red grouper stock dynamics.

*Table 3.1. General definitions, input data, population model, and negative log-likelihood components of the statistical catch-at-age model applied to red grouper. 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 | $y$ | $y \in \{1976 \ldots 2008\}$ |
| Index of ages | $a$ | $a \in \{1 \ldots A\}$, where $A = 16^{+}$ |
| Index of size-limit periods | $r$ | $r \in \{1 \ldots 3\}$ where $1 = 1976 - 1983$ (no size limit), $2 = 1984 - 1991$ (12-inch TL limit), and $3 = 1992 - 2008$ (20-inch limit) |
| Index of length bins | $l$ | $l \in \{1 \ldots 45\}$ |
| Length bins | $l'$ | $l' \in \{160, 190, \ldots, 1510\text{mm}\}$, with midpoint of 30mm bin used to match length compositions. Largest 11 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 other, 3=recreational headboat, 4=general recreational (MRFSS) |
| Index of discards | $d$ | $d \in \{1 \ldots 3\}$ where 1=commercial lines, 2=recreational headboat, 3=general recreational (MRFSS) |
| Index of CPUE | $u$ | $u \in \{1 \ldots 5\}$ where 1 = commercial lines, 2 = headboat, 3 = MRFSS, 4 = FL Keys Visual Survey, 5 = MARMAP chevron trap |
| **Input Data** | | |
| Proportion female at age | $\rho_a$ | Monotonically decreasing; determined by DW; assumed constant across years |
| Proportion females mature at age | $m_a$ | Logistic increase with age; determined by DW; assumed constant across years |
| Spawning date | $t_{\text{spawn}}$ | Fraction denoting the proportional time of year when spawning occurs. Set to 0.315 for red grouper by assuming peak spawning occurs in mid-April. |
| 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 $u$ (often set to number of collection events due to non-independence of samples) |
| 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 fishery 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 |

*Table 3.1.* (continued)

| Quantity | Symbol | Description or definition |
|---|---|---|
| Observed abundance indices | $U_{u,y}$ | $u=1$, commercial lines (weight), $y \in \{1993 \ldots 2008\}$<br>$u=2$, headboat (numbers), $y \in \{1978 \ldots 2008\}$<br>$u=3$, MRFSS (numbers), $y \in \{1991 \ldots 2008\}$<br>$u=4$, FL Keys visual survey (numbers), $y \in \{1994 \ldots 2008\}$<br>$u=5$, MARMAP chevron trap (numbers), $y \in \{1990 \ldots 2008\}$ |
| CVs of abundance indices | $c^U_{u,y}$ | $u=\{1 \ldots 5\}$ as above. Annual values estimated from delta-lognormal GLM for $u=1,2,3,5$, and from knowledge of sampling design for $u=4$ Each time series was scaled to a mean of 1.0 |
| Natural mortality rate | $M_a$ | Function of weight at age ($w_a$): $M_a = \alpha w_a^{\beta}$, with estimates of $\alpha$ and $\beta$ 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 | $\delta_d$ | Proportion discards by fishery $d$ that die. The DW recommended $\delta_d = 0.2$ for all fisheries. |
| 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 |
| **Population Model** | | |
| Mean length at age | $l_a$ | Total length (midyear); $l_a = L_\infty(1-\exp[-K(a-t_0+0.5)])$<br>where $K$, $L_\infty$, and $t_0$ are parameters estimated by the DW |
| CV of $l_a$ | $\widehat{c}^{\lambda}_a$ | Estimated variation of growth, assumed constant across ages. |
| Age–length conversion of population | $\psi^u_{a,l}$ | $\psi^u_{a,l} = \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a l_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a l_a)^2\right)}$ , the Gaussian density function.<br>Matrix $\psi^u$ 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 | $\psi^L_{f,a,l,y}$ | $\psi^L_{f,a,l,y} = \begin{cases} \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a l_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a l_a)^2\right)} & : l_a \geq l_{\text{limit}} \\ 0 & : \text{otherwise} \end{cases}$<br>where $l_{\text{limit}}$ 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 11 length bins fit as a plus group. |
| Age–length conversion of discards | $\psi^D_{d,a,l,y}$ | $\psi^D_{d,a,l,y} = \begin{cases} \frac{1}{\sqrt{2\pi}(\widehat{c}^{\lambda}_a l_a)} \frac{\exp\left[-\left(l'_l - l_a\right)^2\right]}{\left(2(\widehat{c}^{\lambda}_a l_a)^2\right)} & : l_a < l_{\text{limit}} \\ 0 & : \text{otherwise} \end{cases}$<br>where $l_{\text{limit}}$ is the size limit for fishery $d$ in year $y$ (and could be treated as $\infty$ 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 $\psi^L_{f,\cdot,\cdot,y}$ for landings or $\psi^D_{d,\cdot,\cdot,y}$ for discards |
| Individual weight at age of population | $w_a$ | Computed from length at age by<br>$w_a = \theta_1 l_a^{\theta_2}$<br>where $\theta_1$ and $\theta_2$ are parameters from the DW |

*Table 3.1.* (continued)

| **Quantity** | **Symbol** | **Description or definition** |
|---|---|---|
| Individual weight at age of landings and discards | $w^{L,D}_{(f,d),a,y}$ | Computed from length at age by $w^{L,D}_{(f,d),a,y} = \theta_1(\xi^{L,D}_{(f,d),a,y})^{\theta_2}$ |
| Fishery and index selectivities | $s_{(f,u),a,r}$ | $s_{f,a,r} = \begin{cases} \frac{1}{1+\exp[-\widehat{\eta}_{1,f,r}(a-\widehat{\alpha}_{1,f,r})]} & :f=1,3,4; u=4 \\ \left(\frac{1}{\max s_{(f,u),a,r}}\right)\left(\frac{1}{1+\exp\left[-\widehat{\eta}_{1,(f,u),r}\left(a-\widehat{\alpha}_{1,(f,u),r}\right)\right]}\right) \left(1-\frac{1}{1+\exp\left[-\widehat{\eta}_{2,(f,u),r}\left(a-\left[\widehat{\alpha}_{1,(f,u),r}+\widehat{\alpha}_{2,(f,u),r}\right]\right)\right]}\right) & :f=2; u=5 \end{cases}$ where $\widehat{\eta}_{1,(f,u),r}$, $\widehat{\eta}_{2,(f,u),r}$, $\widehat{\alpha}_{1,(f,u),r}$, and $\widehat{\alpha}_{2,(f,u)),r}$ are estimated parameters. Not all parameters were estimated for each fishery (or index) and each period of regulations; some parameters were fixed as described in the text. For instance, commercial hand line selectivity for the first regulatory period was set to commercial hand line selectivity in the second period. Likewise, general recreational selectivity in the first two regulatory periods was set to estimates obtained for the headboat fishery. For the RVC index, a declining logistic model was specified, with 100% selection at the first age class (the slope of the logistic decline in selectivity was estimated). |
| Discard selectivity | $s'_{d,a,r}$ | $s'_{d,1,r}$ estimated; $s'_{d,2,r}$ set to 1.0; $s'_{d,3^+,r}$ set equal to the age-specific probability of total length below the size limit in period $r$. All fisheries with observed discards were assumed to have the same discard selectivity. |
| Fishing mortality rate of landings | $F_{f,a,y}$ | $F_{f,a,y} = s_{f,a,y}\widehat{F}_{f,y}$ where $\widehat{F}_{f,y}$ is an estimated fully selected fishing mortality rate by fishery and $s_{f,a,y} = s_{f,a,r}$ for $y$ in the years represented by $r$ |
| Fishing mortality rate of discards | $F^D_{d,a,y}$ | $F^D_{d,a,y} = s'_{d,a,r}\widehat{F}^D_{d,y}$ where $\widehat{F}^D_{d,y}$ is an estimated fully selected fishing mortality rate of discards by fishery |
| Total fishing mortality rate | $F_{a,y}$ | $F_{a,y} = \sum_f F_{f,a,y} + \sum_d F^D_{d,a,y}$ |
| Total mortality rate | $Z_{a,y}$ | $Z_{a,y} = M_a + F_{a,y}$ |
| Apical F | $F_y$ | $F_y = \max(F_{a,y})$ |
| Abundance at age | $N_{a,y}$ | $N_{1,1976} = \frac{0.8\widehat{R}_0\widehat{h}S_{equil}}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_{1976})$ $\widehat{N}_{2+,1976}$ estimated subject to penalties for deviating from equilibrium conditions expected given assumptions about initial fishing mortality (see "Objective Function") $N_{1,y+1} = \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_{y+1})\text{for } y \geq 1976$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (1\ldots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ 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 $\varsigma = \exp(\sigma^2/2)$, where $\sigma^2$ is the variance of recruitment deviations. Quantities $\phi_0$, $S_y$, and $S_{equil}$ are described below. |

*Table 3.1.* (continued)

| Quantity | Symbol | Description or definition |
|---|---|---|
| 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 mid-April 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 | $NPR_a$ | $NPR_1 = 1 \times \exp(-t_{\text{spawn}}M_1)$ $NPR_{a+1} = NPR_a\exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (1 \ldots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_A t_{\text{spawn}})]}{1-\exp(-M_A)}$ |
| Unfished spawning biomass per recruit | $\phi_0$ | $\phi_0 = NPR_1 w_1 \rho_1 m_1 + \sum_{a=2}^{A} NPR_a w_a(\rho_a m_a + (1-\rho_a))$ |
| Spawning biomass | $S_y$ | $S_y = N''_{1,y} w_1 \rho_1 m_1 + \sum_{a=2}^{A} N''_{a,y} w_a(\rho_a m_a + (1-\rho_a))$ Also referred to as spawning stock biomass in units of total mature biomass (males + females). All males greater than one year of age were assumed mature. |
| Initialization mortality at age | $Z_a^{init}$ | $Z_a^{init} = M_a + \exp(\log(F_{a,1976}) + \log(F_{a,1977}) + \log(F_{a,1978}))/3$ |
| Initial equilibrium abundance at age | $N_a^{equil}$ | Obtained using methods of Caswell (2001) assuming $Z_a^{init}$ as a constant force of mortality |
| Initial equilibrium spawning biomass | $S_{equil}$ | $S_{equil} = N_a^{equil} w_1 \rho_1 m_1 + \sum_{a=2}^{A} N_a^{equil} w_a(\rho_a m_a + (1-\rho_a))$ |
| Population biomass | $B_y$ | $B_y = \sum_a N_{a,y} w_a$ |
| Landing at age in numbers | $L'_{f,a,y}$ | $L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}} N_{a,y}[1-\exp(-Z_{a,y})]$ |
| Landing at age in weight | $L''_{f,a,y}$ | $L''_{f,a,y} = w^L_{f,a,y} L'_{f,a,y}$ |
| Discard mortalities at age in numbers | $D'_{d,a,y}$ | $D'_{d,a,y} = \frac{F^D_{d,a,y}}{Z_{a,y}} N_{a,y}[1-\exp(-Z_{a,y})]$ |
| Discard mortalities at age in weight | $D''_{d,a,y}$ | $D''_{d,a,y} = w^D_{d,a,y} D'_{d,a,y}$ |

*Table 3.1.* (continued)

| **Quantity** | **Symbol** | **Description or definition** |
|---|---|---|
| Index catchability | $q_{u,y}$ | $q_{u,1976} = \widehat{q}^0_u f(\text{density})$<br>$q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ for $y \geq 1976$<br>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 up until 2003, where technology effects were thought to saturate (see SEDAR 19 DW report):<br>$f_y(\text{trend}) = \begin{cases} 1.0 & :y = 1976 \\ f_{y-1}(\text{trend}) * (\text{y} - 1976)\beta_{\text{q}} & :1976 < y \leq 2003 \\ f_{2003}(\text{trend}) & :2003 < y \end{cases}.$<br>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, only 1-2 of the three catchability function may be used at any one time to achieve parameter identifiability. Catchability from fishery-independent surveys was assumed constant. |
| 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 | $\breve{U}_{u,y}$ | $\breve{U}_{u,y} = \begin{cases} \widehat{q}_{u,y} \sum_a w^L_{f=u,a,y} N'_{a,y} s_{u,a,r} & : \quad u = 1 \\ \widehat{q}_{u,y} \sum_a N'_{a,y} s_{u,a,r} & : \quad u = 2, 3, 4, 5 \end{cases}$<br>where $s_{u,a,r}$ is the selectivity of the relevant fishery in the year corresponding to $y$. Selectivity for the general recreational index included landed and discarded fish. |
| **Objective Function** | | |
| Multinomial length compositions | $\Lambda_1$ | $\Lambda_1 = -\omega_1 \sum_{f,d,u} \sum_y \left[ n^{\lambda}_{(f,d,u),y} \sum_l (p^{\lambda}_{(f,d,u),l,y} + x) \log \left( \frac{(\breve{p}^{\lambda}_{(f,d,u),l,y} + x)}{(p^{\lambda}_{(f,d,u),l,y} + x)} \right) \right]$<br>where $\omega_1$ is a preset weight 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. |

*Table 3.1.* (continued)

| **Quantity** | **Symbol** | **Description or definition** |
|---|---|---|
| Multinomial age compositions | $\Lambda_2$ | $\Lambda_2 = -\omega_2 \sum_{f,u} \sum_{y} \left[ n^{\alpha}_{(f,u),y} \sum_{a} (p^{\alpha}_{(f,u),a,y} + x) \log \left( \frac{(\breve{p}^{\alpha}_{(f,u),a,y} + x)}{(p^{\alpha}_{(f,u),a,y} + x)} \right) \right]$ where $\omega_2$ is a preset weight and $x$ =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term. |
| Lognormal landings | $\Lambda_3$ | $\Lambda_3 = \omega_3 \sum_{f} \sum_{y} \frac{\left[ \log \left( (L_{f,y} + x) \big/ (\breve{L}_{f,y} + x) \right) \right]^2}{2(\sigma^L_{f,y})^2}$ where $\omega_3$ is a preset weight and $x$ =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^L_{f,y} = \sqrt{\log(1 + (c^L_{f,y})^2)}$. |
| Lognormal discard mortalities | $\Lambda_4$ | $\Lambda_4 = \omega_4 \sum_{d} \sum_{y} \frac{\left[ \log \left( (\delta_d D_{d,y} + x) \big/ (\breve{D}_{d,y} + x) \right) \right]^2}{2(\sigma^D_{d,y})^2}$ where $\omega_4$ is a preset weight and $x$ =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^D_{d,y} = \sqrt{\log(1 + (c^D_{d,y})^2)}$. |
| Lognormal CPUE | $\Lambda_5$ | $\Lambda_5 = \sum_{u} \omega_5 \sum_{y} \frac{\left[ \log \left( (U_{u,y} + x) \big/ (\breve{U}_{u,y} + x) \right) \right]^2}{2(\sigma^U_{u,y})^2}$ where $\omega_5$ is a preset weight and $x$ =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma^U_{u,y} = \sqrt{\log(1 + (c^U_{u,y})^2)}$. |
| Constraint on recruitment deviations | $\Lambda_6$ | $\Lambda_6 = \omega_6 \left[ R^2_{1976} + \sum_{y>1976} (R_y - \widehat{\varrho} R_{y-1})^2 \right]$ where $R_y$ are recruitment deviations in log space, $\omega_6 = 1$ is a preset weight and $\widehat{\varrho}$ is the estimated first-order autocorrelation |
| Additional constraint on initial recruitment deviation | $\Lambda_7$ | $\Lambda_7 = \omega_7 \left( R^2_{1976} \right)$ where $\omega_7$ is a preset weight |
| Additional constraint on final recruitment deviations | $\Lambda_8$ | $\Lambda_8 = \omega_8 \left( \sum_{y \geq 2007} R_y - \widehat{\varrho} R_{y-1})^2 \right)$ where $\omega_8$ is a preset weight |
| Penalty on random walk on catchability | $\Lambda_9$ | $\Lambda_9 = \omega_9 \sum_{u} \sum_{y} \frac{\epsilon^2_{u,y}}{2(\sigma^q_u)^2}$ where $\omega_9$ is a preset weight and $\sigma^q_u$ is a control variable input by the user defining the standard deviation of the random walk process. As $\sigma^q_u$ increases, one essentially estimates each deviation as a free parameter, while values close to zero allow little variation in annual catchability. |
| Penalty on initial age structure | $\Lambda_{10}$ | $\Lambda_{10} = \sum_{a=2}^{A} (\omega_{10} (\widehat{N}_{a,1976} - N^{equil}_a)^2$ where $\omega_{10}$ is a preset weight. |
| Prior distributions and penalties | $\Lambda_{11}$ | Several diffuse prior distributions were imposed on parameters to keep them in reasonable parameter spaces. This included $\widehat{h}$, $\widehat{\rho_R}$, $\widehat{c^{\lambda}_a}$, $\widehat{\psi}$, and $\widehat{\eta}_{1,(f,u),r}$: $\Lambda_{11} = \frac{(\widehat{h} - E(h))^2}{\sigma^2_h} + \frac{(\widehat{c^{\lambda}_a} - E(c^{\lambda}_a))^2}{\sigma^2_c} + (\rho_R - E(\rho_R))^2 + \sum_{f,u} (\widehat{\eta}_{1,(f,u),r} - E(\eta_{1,(f,u),r}))^2$, where expected values $E(\theta)$ are supplied by the analyst. |
| Total objective function | $\Lambda$ | $\Lambda = \sum_{i=1}^{11} \Lambda_i$ Objective function minimized by the assessment model |

## References

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. In Review. When can we reliably estimate the productivity of fish stocks? Canadian Journal of Fisheries and Aquatic Sciences .

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.

Ltd, O. R., 2004. An Introduction to AD Model Builder version 7.1.1. Box 2040, Sidney, British Columbia.

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, 2009b. SEDAR-19-DW-06: Steepness of spawner-recruit relationships in reef fishes of the southeastern U.S.: A prior distribution for possible use in stock assessment.

Appendix A: AD Model Builder code for estimation

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR19 Assessment: Red Grouper, August 2009
//##
//##  Kyle Shertzer, NMFS, Beaufort Lab
//##  Kyle.Shertzer@noaa.gov
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

DATA_SECTION
//Create ascii file for output
//!!CLASS ofstream report1("rgresults.rep",ios::out);  //create file for output

!!cout << "Starting Red Grouper Assessment Model" << endl;

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

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

//3 periods of size regs: styr-83 no restrictions, 1984-91 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;  //16 inch limit in mm (splits difference between FL and fed regs for prior to period 3)

//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;
init_int nlenbins_plus;
// 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);

//discard mortality constants
init_number set_Dmort_cL;
init_number set_Dmort_HB;
init_number set_Dmort_MRFSS;

//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;
// Von Bert parameters in TL mm
init_number set_Linf;
init_number set_K;
init_number set_t0;
//Standard erros of von bert params
init_number set_Linf_se;
init_number set_K_se;
init_number set_t0_se;
//CV of length at age and its standard error
init_number set_len_cv;
init_number set_len_cv_se;
//TL(mm)-weight(whole weight in g) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;
//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector maturity_m_obs(1,nages);            //proportion males 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
//Spawner-recruit parameters (Initial guesses or fixed values)
init_number set_steep;
init_number set_steep_se;
init_number set_log_R0;
init_number set_R_autocorr;

//##############################################################################
//###FL Keys Visual Survey#################################
//CPUE
init_int styr_RVC_cpue;
init_int endyr_RVC_cpue;
init_vector obs_RVC_cpue(styr_RVC_cpue,endyr_RVC_cpue);   //Observed CPUE
init_vector RVC_cpue_cv(styr_RVC_cpue,endyr_RVC_cpue);    //CV of cpue
//Length Compositions (3 cm bins)
init_int styr_RVC_lenc;
init_int endyr_RVC_lenc;
init_vector nsamp_RVC_lenc(styr_RVC_lenc,endyr_RVC_lenc);
init_vector neff_RVC_lenc(styr_RVC_lenc,endyr_RVC_lenc);
init_matrix obs_RVC_lenc(styr_RVC_lenc,endyr_RVC_lenc,1,nlenbins);
//###############MARMAP Chevron trap index################################
//CPUE
init_int styr_CVT_cpue;
init_int endyr_CVT_cpue;
init_vector obs_CVT_cpue(styr_CVT_cpue,endyr_CVT_cpue);   //Observed CPUE
init_vector CVT_cpue_cv(styr_CVT_cpue,endyr_CVT_cpue);    //cv of cpue
//Length Compositions (3cm bins)
init_int styr_CVT_lenc;
init_int endyr_CVT_lenc;
init_vector nsamp_CVT_lenc(styr_CVT_lenc,endyr_CVT_lenc);
init_vector neff_CVT_lenc(styr_CVT_lenc,endyr_CVT_lenc);
init_matrix obs_CVT_lenc(styr_CVT_lenc,endyr_CVT_lenc,1,nlenbins);
//Age Compositions
init_int styr_CVT_agec;
init_int endyr_CVT_agec;
init_vector nsamp_CVT_agec(styr_CVT_agec,endyr_CVT_agec);
init_vector neff_CVT_agec(styr_CVT_agec,endyr_CVT_agec);
init_matrix obs_CVT_agec(styr_CVT_agec,endyr_CVT_agec,1,nages);

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

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

// Discards (1000 fish)
init_int styr_cL_D;
init_int endyr_cL_D;
init_vector obs_cL_released(styr_cL_D,endyr_cL_D); //vector of observed releases by year, 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 styr_cL_lenc;
init_int endyr_cL_lenc;
init_vector nsamp_cL_lenc(styr_cL_lenc,endyr_cL_lenc);
init_vector neff_cL_lenc(styr_cL_lenc,endyr_cL_lenc);
init_matrix obs_cL_lenc(styr_cL_lenc,endyr_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);

//#############################################################################
//##Commercial Other gears (pots + dive + other) fishery
// Landings (1000 lb whole weight)
init_int styr_cO_L;
init_int endyr_cO_L;
init_vector obs_cO_L(styr_cO_L,endyr_cO_L);
init_vector cO_L_cv(styr_cO_L,endyr_cO_L);    //vector of CV of landings by year
// Length Compositions (3 cm bins, data from pots)
init_int nyr_cO_lenc;
init_ivector yrs_cO_lenc(1,nyr_cO_lenc);
init_vector nsamp_cO_lenc(1,nyr_cO_lenc);
init_vector neff_cO_lenc(1,nyr_cO_lenc);
init_matrix obs_cO_lenc(1,nyr_cO_lenc,1,nlenbins);


//#############################################################################
//################################Headboat 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 styr_HB_lenc;
init_int endyr_HB_lenc;
init_vector nsamp_HB_lenc(styr_HB_lenc,endyr_HB_lenc);
init_vector neff_HB_lenc(styr_HB_lenc,endyr_HB_lenc);
init_matrix obs_HB_lenc(styr_HB_lenc,endyr_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 discards
init_int styr_HB_D_lenc;
init_int endyr_HB_D_lenc;
init_vector nsamp_HB_D_lenc(styr_HB_D_lenc,endyr_HB_D_lenc);
init_vector neff_HB_D_lenc(styr_HB_D_lenc,endyr_HB_D_lenc);
init_matrix obs_HB_D_lenc(styr_HB_D_lenc,endyr_HB_D_lenc,1,nlenbins);

//#############################################################################
//############################MRFSS landings #################################
//CPUE
init_int styr_MRFSS_cpue;
init_int endyr_MRFSS_cpue;
init_vector obs_MRFSS_cpue(styr_MRFSS_cpue,endyr_MRFSS_cpue);//Observed CPUE
init_vector MRFSS_cpue_cv(styr_MRFSS_cpue,endyr_MRFSS_cpue); //CV of cpue
// Landings (1000 fish)
init_int styr_MRFSS_L;
init_int endyr_MRFSS_L;
init_vector obs_MRFSS_L(styr_MRFSS_L,endyr_MRFSS_L);
init_vector MRFSS_L_cv(styr_MRFSS_L,endyr_MRFSS_L);
// Discards (1000s)
init_int styr_MRFSS_D;
init_int endyr_MRFSS_D;
init_vector obs_MRFSS_released(styr_MRFSS_D,endyr_MRFSS_D); //vector of observed releases by year, multiplied by discard mortality for fitting
init_vector MRFSS_D_cv(styr_MRFSS_D,endyr_MRFSS_D);    //vector of CV of discards by year
// Length Compositions (3 cm bins)
init_int styr_MRFSS_lenc;
init_int endyr_MRFSS_lenc;
init_vector nsamp_MRFSS_lenc(styr_MRFSS_lenc,endyr_MRFSS_lenc);
init_vector neff_MRFSS_lenc(styr_MRFSS_lenc,endyr_MRFSS_lenc);
init_matrix obs_MRFSS_lenc(styr_MRFSS_lenc,endyr_MRFSS_lenc,1,nlenbins);
// Age Compositions
init_int styr_MRFSS_agec;
init_int endyr_MRFSS_agec;
init_vector nsamp_MRFSS_agec(styr_MRFSS_agec,endyr_MRFSS_agec);
init_vector neff_MRFSS_agec(styr_MRFSS_agec,endyr_MRFSS_agec);
```

```
init_matrix obs_MRFSS_agec(styr_MRFSS_agec,endyr_MRFSS_agec,1,nages);

//##############################################################################
//##################Parameter values and initial guesses #################################
//Initial guesses of estimated selectivity parameters
init_number set_selpar_L50_RVC;
init_number set_selpar_slope_RVC;
init_number set_selpar_L502_RVC;
init_number set_selpar_slope2_RVC;

init_number set_selpar_L50_CVT;
init_number set_selpar_slope_CVT;
init_number set_selpar_L502_CVT;
init_number set_selpar_slope2_CVT;

//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_L502_cL2;
init_number set_selpar_slope2_cL2;
init_number set_selpar_L50_cL3;
init_number set_selpar_slope_cL3;
init_number set_selpar_L502_cL3;
init_number set_selpar_slope2_cL3;

//init_vector set_sel_cL_D_2(1,nages);
//init_number set_selpar_Age1_cL_D2;
//init_number set_selpar_Age2_cL_D2;
////init_number set_selpar_L50_cL_D2;
////init_number set_selpar_slope_cL_D2;
////init_number set_selpar_L502_cL_D2;
////init_number set_selpar_slope2_cL_D2;

//init_number set_selpar_L50_cO1;
//init_number set_selpar_slope_cO1;
//init_number set_selpar_L502_cO1;
//init_number set_selpar_slope2_cO1;
init_number set_selpar_L50_cO2;
init_number set_selpar_slope_cO2;
init_number set_selpar_L502_cO2;
init_number set_selpar_slope2_cO2;
init_number set_selpar_L50_cO3;
//init_number set_selpar_slope_cO3;
//init_number set_selpar_L502_cO3;
//init_number set_selpar_slope2_cO3;

init_number set_selpar_L50_HB1;
init_number set_selpar_slope_HB1;
init_number set_selpar_L502_HB1;
init_number set_selpar_slope2_HB1;
init_number set_selpar_L50_HB2;
init_number set_selpar_slope_HB2;
init_number set_selpar_L502_HB2;
init_number set_selpar_slope2_HB2;
init_number set_selpar_L50_HB3;
init_number set_selpar_slope_HB3;
init_number set_selpar_L502_HB3;
init_number set_selpar_slope2_HB3;


//init_vector set_sel_HB_D_2(1,nages);
//init_vector set_sel_HB_D_3(1,nages);
init_number set_selpar_Age1_HB_D3;
//init_number set_selpar_Age2_HB_D3;
//init_number set_selpar_L50_HB_D3;
//init_number set_selpar_slope_HB_D3;
//init_number set_selpar_L502_HB_D3;
//init_number set_selpar_slope2_HB_D3;

init_number set_selpar_L50_MRFSS3;
init_number set_selpar_slope_MRFSS3;
init_number set_selpar_L502_MRFSS3;
init_number set_selpar_slope2_MRFSS3;

//--weights for likelihood components-------------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_D;
init_number set_w_lc;
init_number set_w_ac;
init_number set_w_I_RVC;
init_number set_w_I_CVT;
init_number set_w_I_cL;
init_number set_w_I_HB;
init_number set_w_I_MRFSS;
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 (MRFSS) landings multiplicative bias
init_number set_L_mrfss_bias;

////--index catchability-------------------------------------------------------------------------------------------------------------
init_number set_logq_RVC;     //catchability coefficient (log) for MARMAP RVC
init_number set_logq_CVT;     //catchability coefficient (log) for MARMAP CVT
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
init_number set_logq_MRFSS;   //catchability coefficient (log) for MRFSS CPUE index
```

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

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

////--F's--------------------------------
init_number set_log_avg_F_cL;
init_number set_log_avg_F_cO;
init_number set_log_avg_F_HB;
init_number set_log_avg_F_MRFSS;
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_MRFSS_D;
init_number set_F_cL_D_ratio;  //early F on commercial discards as a ratio of later F
init_number set_F_HB_D_ratio;  //early F on HB D as a ratio of later F

//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_RVC_lenc;
init_number minSS_CVT_lenc;
init_number minSS_cL_lenc;
init_number minSS_cO_lenc;
init_number minSS_HB_lenc;
init_number minSS_HB_D_lenc;
init_number minSS_MRFSS_lenc;

//threshold sample sizes for age comps
init_number minSS_CVT_agec;
init_number minSS_cL_agec;
init_number minSS_HB_agec;
init_number minSS_MRFSS_agec;

//ageing error matrix (columns are true ages, rows are ages as read for age comps)
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_cO2;
init_number set_p_lenc_cO3;
init_number set_p_lenc_HB2;
init_number set_p_lenc_HB3;
init_number set_p_lenc_MRFSS2;
init_number set_p_lenc_MRFSS3;

init_number set_p_lenc_cL_D2;
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_MRFSS_D2;
init_number set_p_lenc_MRFSS_D3;

// #######Indexing integers for year(iyear), age(iage),length(ilen) ###############
int iyear;
int iage;
int ilen;
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(-3.0,0.0,2);
  number Linf;
  number K;
  number t0;
  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

  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_cO_mm(styr,endyr,1,nages);       //mean length at age of cO landings in mm (may differ from popn mean)
  matrix wgt_cO_klb(styr,endyr,1,nages);      //whole wgt of cO landings in 1000 lb
  matrix len_HB_mm(styr,endyr,1,nages);         //mean length at age of HB landings in mm (may differ from popn mean)
  matrix wgt_HB_klb(styr,endyr,1,nages);        //whole wgt of HB landings in 1000 lb
  matrix len_MRFSS_mm(styr,endyr,1,nages);       //mean length at age of MRFSS landings in mm (may differ from popn mean)
  matrix wgt_MRFSS_klb(styr,endyr,1,nages);      //whole wgt of MRFSS 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_MRFSS_D_mm(styr,endyr,1,nages);      //mean length at age of cL discards in mm (may differ from popn mean)
  matrix wgt_MRFSS_D_klb(styr,endyr,1,nages);     //whole wgt of cL discards in 1000 lb


  vector meanlen_TL(1,nages);        //mean Total length (mm) at age

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

  matrix lenprob_cL1(1,nages,1,nlenbins);     //distn of size at age in cL period 1
  matrix lenprob_cL2(1,nages,1,nlenbins);     //distn of size at age in cL period 2
  matrix lenprob_cL3(1,nages,1,nlenbins);     //distn of size at age in cL period 2
  matrix lenprob_cO1(1,nages,1,nlenbins);     //distn of size at age in cL period 1
  matrix lenprob_cO2(1,nages,1,nlenbins);     //distn of size at age in cL period 2
  matrix lenprob_cO3(1,nages,1,nlenbins);     //distn of size at age in cL period 3
  matrix lenprob_HB1(1,nages,1,nlenbins);       //distn of size at age in HB period 1
  matrix lenprob_HB2(1,nages,1,nlenbins);       //distn of size at age in HB period 2
  matrix lenprob_HB3(1,nages,1,nlenbins);       //distn of size at age in HB period 3
  matrix lenprob_MRFSS1(1,nages,1,nlenbins);       //distn of size at age in MRFSS period 1
  matrix lenprob_MRFSS2(1,nages,1,nlenbins);       //distn of size at age in MRFSS period 2
  matrix lenprob_MRFSS3(1,nages,1,nlenbins);       //distn of size at age in MRFSS period 3

  matrix lenprob_cL_D2(1,nages,1,nlenbins);     //distn of size at age in cL discards comm period 2
  matrix lenprob_cL_D3(1,nages,1,nlenbins);     //distn of size at age in cL discards comm period 3
  matrix lenprob_HB_D2(1,nages,1,nlenbins);     //distn of size at age in HB discards rec period 2. not used for comp data, just avg wgt.
  matrix lenprob_HB_D3(1,nages,1,nlenbins);     //distn of size at age in HB discards rec period 3
  matrix lenprob_MRFSS_D2(1,nages,1,nlenbins);     //distn of size at age in HB discards rec period 2. not used for comp data, just avg wgt.
  matrix lenprob_MRFSS_D3(1,nages,1,nlenbins);     //distn of size at age in HB discards rec period 3

  init_bounded_number log_len_cv(-3.0,-0.5,4);
//  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
  vector len_cv(1,nages);

//  number age_limit_12in; //age corresponding to 12-inch size limit, given mean growth curve
//  number age_limit_20in; //age corresponding to 20-inch size limit, given mean growth curve

////----Predicted length and age compositions
  matrix pred_RVC_lenc(styr_RVC_lenc,endyr_RVC_lenc,1,nlenbins);
  matrix pred_CVT_lenc(styr_CVT_lenc,endyr_CVT_lenc,1,nlenbins);
  matrix pred_cL_lenc(styr_cL_lenc,endyr_cL_lenc,1,nlenbins);
  matrix pred_cO_lenc(1,nyr_cO_lenc,1,nlenbins);
  matrix pred_HB_lenc(styr_HB_lenc,endyr_HB_lenc,1,nlenbins);
  matrix pred_HB_D_lenc(styr_HB_D_lenc,endyr_HB_D_lenc,1,nlenbins);
  matrix pred_MRFSS_lenc(styr_MRFSS_lenc,endyr_MRFSS_lenc,1,nlenbins);

  //##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_cO(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_MRFSS2(0.0,1.0,3);
  //init_bounded_number p_lenc_MRFSS3(0.0,1.0,3);
  number p_lenc_cL2;
  number p_lenc_cL3;
  number p_lenc_cO2;
  number p_lenc_cO3;
  number p_lenc_HB2;
  number p_lenc_HB3;
  number p_lenc_MRFSS2;
  number p_lenc_MRFSS3;

  //init_bounded_number p_lenc_HB_D3(0.0,1.0,3);
  number p_lenc_cL_D2;
  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_MRFSS_D2; //no comp data in this period, this par only used for avg weight
  number p_lenc_MRFSS_D3;

  matrix pred_CVT_agec(styr_CVT_agec,endyr_CVT_agec,1,nages);
  matrix ErrorFree_CVT_agec(styr_CVT_agec,endyr_CVT_agec,1,nages); //age comps prior to applying ageing error matrix
  matrix pred_cL_agec(1,nyr_cL_agec,1,nages);
  matrix ErrorFree_cL_agec(1,nyr_cL_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_MRFSS_agec(styr_MRFSS_agec,endyr_MRFSS_agec,1,nages);
  matrix ErrorFree_MRFSS_agec(styr_MRFSS_agec,endyr_MRFSS_agec,1,nages);

  //nsamp_X_allyr vectors used only for R output of comps with nonconsecutive yrs, given sample size cutoffs
  vector nsamp_RVC_lenc_allyr(styr,endyr);
  vector nsamp_CVT_lenc_allyr(styr,endyr);
  vector nsamp_cL_lenc_allyr(styr,endyr);
  vector nsamp_cO_lenc_allyr(styr,endyr);
  vector nsamp_HB_lenc_allyr(styr,endyr);
  vector nsamp_HB_D_lenc_allyr(styr,endyr);
  vector nsamp_MRFSS_lenc_allyr(styr,endyr);
  vector nsamp_CVT_agec_allyr(styr,endyr);
  vector nsamp_cL_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);
  vector nsamp_MRFSS_agec_allyr(styr,endyr);

//effective sample size applied in multinomial distributions
  vector neff_RVC_lenc_allyr(styr,endyr);
  vector neff_CVT_lenc_allyr(styr,endyr);
  vector neff_cL_lenc_allyr(styr,endyr);
  vector neff_cO_lenc_allyr(styr,endyr);
  vector neff_HB_lenc_allyr(styr,endyr);
  vector neff_HB_D_lenc_allyr(styr,endyr);
  vector neff_MRFSS_lenc_allyr(styr,endyr);
  vector neff_CVT_agec_allyr(styr,endyr);
  vector neff_cL_agec_allyr(styr,endyr);
  vector neff_HB_agec_allyr(styr,endyr);
  vector neff_MRFSS_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_RVC_lenc_allyr_out(styr,endyr);
  vector neff_CVT_lenc_allyr_out(styr,endyr);
  vector neff_cL_lenc_allyr_out(styr,endyr);
  vector neff_cO_lenc_allyr_out(styr,endyr);
  vector neff_HB_lenc_allyr_out(styr,endyr);
  vector neff_HB_D_lenc_allyr_out(styr,endyr);
  vector neff_MRFSS_lenc_allyr_out(styr,endyr);
  vector neff_CVT_agec_allyr_out(styr,endyr);
  vector neff_cL_agec_allyr_out(styr,endyr);
  vector neff_HB_agec_allyr_out(styr,endyr);
  vector neff_MRFSS_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
  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 maturity_m(1,nages);            //Proportion of female mature at age
  vector reprod(1,nages);
//
////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(11,15,1);   //log(virgin Recruitment)
  //number log_R0;
  number R0;                             //virgin recruitment
  init_bounded_number steep(0.21,0.99,3); //steepness
//  number steep;  //uncomment to fix steepness, comment line directly above
  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr,-3,3,2); //log recruitment deviations
  //vector log_rec_dev(styr_rec_dev,endyr);
  vector log_rec_dev_output(styr,endyr+1);        //used in output. equals zero except for yrs in log_rec_dev
  number var_rec_dev;                    //variance of log recruitment deviations
                                           //Estimate from yrs with unconstrainted S-R(XXXX-XXXX)
  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-------------------------------------------------------------------------
//RVC Survey-------------------------------------------------
  matrix sel_RVC(styr,endyr,1,nages);
  //init_bounded_number selpar_L50_RVC(0.1,8.0,1);
  init_bounded_number selpar_slope_RVC(0.1,10.0,1);
  //init_bounded_number selpar_L502_RVC(1.0,19.0,3); //additive with L50
  //init_bounded_number selpar_slope2_RVC(0.0,12.0,3);
  number selpar_L50_RVC;
//  number selpar_slope_RVC;
  number selpar_L502_RVC; //additive with L50
  number selpar_slope2_RVC;

  vector sel_RVC_vec(1,nages);

//MARMAP CVT-------------------------------------------------
  matrix sel_CVT(styr,endyr,1,nages);
  init_bounded_number selpar_L50_CVT(0.1,8.0,1);
  init_bounded_number selpar_slope_CVT(0.5,19.0,1);
//  number selpar_slope_CVT;
  init_bounded_number selpar_L502_CVT(1.0,6.0,1);
  init_bounded_number selpar_slope2_CVT(0.0,12.0,1);
  vector sel_CVT_vec(1,nages);

//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
  //number selpar_slope_cL2; //period 2
  //number selpar_L50_cL2;
  number selpar_slope2_cL2; //period 2
  number selpar_L502_cL2;

  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_cL3; //period 3
  number selpar_L502_cL3;

  //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 as HB D


//Commercial combined gears-------------------------------------------------
  matrix sel_cO(styr,endyr,1,nages);
  //init_bounded_number selpar_L50_cO1(0.1,8.0,1);
  //init_bounded_number selpar_slope_cO1(0.5,12.0,1);
  //init_bounded_number selpar_L502_cO1(1.0,6.0,3);
  //init_bounded_number selpar_slope2_cO1(0.0,12.0,3);
  //number selpar_L50_cO1;
  //number selpar_slope_cO1;
  //number selpar_L502_cO1;
  //number selpar_slope2_cO1;

  //init_bounded_number selpar_L50_cO2(0.1,8.0,1);
  //init_bounded_number selpar_slope_cO2(0.5,19.0,1);
  //init_bounded_number selpar_L502_cO2(1.0,6.0,3);
  //init_bounded_number selpar_slope2_cO2(0.0,12.0,3);
  number selpar_L50_cO2;
  number selpar_slope_cO2;
  number selpar_L502_cO2;
  number selpar_slope2_cO2;

  number selpar_L50_cO3;
  number selpar_slope_cO3;
  number selpar_L502_cO3;
  number selpar_slope2_cO3;

//  vector sel_cO_1(1,nages); //sel vector
  vector sel_cO_2(1,nages);   //sel vector
  vector sel_cO_3(1,nages);   //sel vector

//Headboat-------------------------------------------------
  matrix sel_HB(styr,endyr,1,nages);
  init_bounded_number selpar_L50_HB1(0.1,8.0,1);
  init_bounded_number selpar_slope_HB1(0.5,12.0,1); //period 1
  //number selpar_slope_HB1; //period 1
  //number selpar_L50_HB1;
  number selpar_slope2_HB1; //period 1
  number selpar_L502_HB1;

  init_bounded_number selpar_L50_HB2(0.1,8.0,1);
  init_bounded_number selpar_slope_HB2(0.5,12.0,1); //period 2
  //number selpar_slope_HB2; //period 2
  //number selpar_L50_HB2;
  number selpar_slope2_HB2; //period 2
  number selpar_L502_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_HB3; //period 3
  number selpar_L502_HB3;

//  //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(0.0,1.0,1); //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
```

```
////MRFSS-------------------------------------------------
  matrix sel_MRFSS(styr,endyr,1,nages);

  init_bounded_number selpar_L50_MRFSS3(0.1,8.0,1);
  init_bounded_number selpar_slope_MRFSS3(0.5,12.0,1); //period 3
//  number selpar_slope_MRFSS3; //period 3
//  number selpar_L50_MRFSS3;
  number selpar_slope2_MRFSS3; //period 3
  number selpar_L502_MRFSS3;

//  //init_bounded_dev_vector selpar_L50_MRFSS_dev(styr_MRFSS_lenc,endyr_period1,-5,5,3);
//  //vector sel_MRFSS_1(1,nages); //sel in period 1
//  //vector sel_MRFSS_2(1,nages); //sel in period 2
  vector sel_MRFSS_3(1,nages); //sel in period 3

  //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_RVC_cpue(styr_RVC_cpue,endyr_RVC_cpue);          //predicted RVC U (fish/trap-hour)
  matrix N_RVC(styr_RVC_cpue,endyr_RVC_cpue,1,nages);          //used to compute RVC index
  vector pred_CVT_cpue(styr_CVT_cpue,endyr_CVT_cpue);          //predicted CVT U (fish/trap-hour)
  matrix N_CVT(styr_CVT_cpue,endyr_CVT_cpue,1,nages);          //used to compute CVT 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
  vector pred_MRFSS_cpue(styr_MRFSS_cpue,endyr_MRFSS_cpue);    //predicted MRFSS U (number/effort)
  matrix N_MRFSS(styr_MRFSS_cpue,endyr_MRFSS_cpue,1,nages);    //used to compute MRFSS index
  matrix sel_MRFSS_cpue(styr,endyr,1,nages);                   //includes discards


////---Catchability (CPUE q's)----------------------------------------------------------
  init_bounded_number log_q_RVC(-20,-5,1);
  init_bounded_number log_q_CVT(-20,-5,1);
  init_bounded_number log_q_cL(-15,-5,1);
  init_bounded_number log_q_HB(-20,-5,1);
  init_bounded_number log_q_MRFSS(-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_MRFSS(styr_MRFSS_cpue,endyr_MRFSS_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_MRFSS(styr_MRFSS_cpue,endyr_MRFSS_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_MRFSS(styr_MRFSS_cpue,endyr_MRFSS_cpue);

//---Landings Bias for recreational landings------------------------------------------------------------------
  //init_bounded_number L_mrfss_bias(0.1,10.0,3);
  number L_mrfss_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_cO_num(styr,endyr,1,nages);              //landings (numbers) at age
  matrix L_cO_klb(styr,endyr,1,nages);              //landings (1000 lb whole weight) at age
  vector pred_cO_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cO_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_MRFSS_num(styr,endyr,1,nages);             //landings (numbers) at age
  matrix L_MRFSS_klb(styr,endyr,1,nages);             //landings (1000 lb whole weight) at age
  vector pred_MRFSS_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_MRFSS_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
//
//---Discards (number dead fish) --------------------------------------------------
  matrix D_cL_num(styr,endyr,1,nages);   //discards (numbers) 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
  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_MRFSS_num(styr,endyr,1,nages);   //discards (numbers) at age
  vector pred_MRFSS_D_knum(styr,endyr);  //yearly discards summed over ages
  vector obs_MRFSS_D(styr_MRFSS_D,endyr_MRFSS_D);        //observed releases multiplied by discard mortality
  vector pred_MRFSS_D_klb(styr,endyr);   //yearly discards in klb summed over ages

  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_cO_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_MRFSS_prop;    //proportion of F_sum attributable to MRFSS, 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_MRFSS_D_prop;  //proportion of F_sum attributable to MRFSS 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;

  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------------------------------------------------------------------
  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_cO(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cO(styr_cO_L,endyr_cO_L,-10.0,5.0,2);
  matrix F_cO(styr,endyr,1,nages);
  vector F_cO_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cO;
  number log_F_dev_end_cO;

  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_MRFSS(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_MRFSS(styr_MRFSS_L,endyr_MRFSS_L,-10.0,5.0,2);
  matrix F_MRFSS(styr,endyr,1,nages);
  vector F_MRFSS_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_MRFSS;
  number log_F_dev_end_MRFSS;

  init_bounded_number F_init_ratio(0.05,2.0,-1);
//--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_avgdev_cL_D;
  number F_cL_D_ratio;  //defines early cL discard F as a proportion of later F
  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_avgdev_HB_D;
  number F_HB_D_ratio;  //defines early hb discard F as a proportion of later F
  number log_F_dev_end_HB_D;

  matrix sel_MRFSS_D(styr,endyr,1,nages);
  init_bounded_number log_avg_F_MRFSS_D(-10.0,0.0,1);
  init_bounded_dev_vector log_F_dev_MRFSS_D(styr_MRFSS_D,endyr_MRFSS_D,-10.0,5.0,2);
  matrix F_MRFSS_D(styr,endyr,1,nages);
  vector F_MRFSS_D_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_MRFSS_D;

  number Dmort_cL;
  number Dmort_HB;
  number Dmort_MRFSS;

//---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;
  number w_ac;
  number w_I_RVC;
  number w_I_CVT;
  number w_I_cL;
  number w_I_HB;
  number w_I_MRFSS;
  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_RVC_cpue;
  number f_CVT_cpue;
  number f_cL_cpue;
  number f_HB_cpue;
  number f_MRFSS_cpue;

  number f_cL_L;
  number f_cO_L;
  number f_HB_L;
  number f_MRFSS_L;

  number f_cL_D;
  number f_HB_D;
  number f_MRFSS_D;

  number f_RVC_lenc;
  number f_CVT_lenc;
  number f_cL_lenc;
  number f_cO_lenc;
  number f_HB_lenc;
  number f_HB_D_lenc;
  number f_MRFSS_lenc;

  number f_CVT_agec;
  number f_cL_agec;
  number f_HB_agec;
  number f_MRFSS_agec;

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

  //Penalties and constraints. Not all are used.
  number f_rec_dev;                //weight on recruitment deviations to fit S-R curve
  number f_rec_dev_early;          //extra weight on deviations in first recruitment stanza
  number f_rec_dev_end;            //extra weight on deviations in first recruitment stanza
  number f_Ftune;                  //penalty for tuning F in Ftune yr.  Not applied in final optimization phase.
  number f_fullF_constraint;       //penalty for Fapex>X
//  number f_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
  objective_function_value fval;
  number fval_unwgt;

//--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_MRFSS=set_Dmort_MRFSS;

  obs_cL_D=Dmort_cL*obs_cL_released;
  obs_HB_D=Dmort_HB*obs_HB_released;
  obs_MRFSS_D=Dmort_MRFSS*obs_MRFSS_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;
  steep=set_steep;
  R_autocorr=set_R_autocorr;

  log_q_RVC=set_logq_RVC;
  log_q_CVT=set_logq_CVT;
  log_q_cL=set_logq_cL;
  log_q_HB=set_logq_HB;
  log_q_MRFSS=set_logq_MRFSS;
  q_rate=set_q_rate;
  q_rate_fcn_cL=1.0;
  q_rate_fcn_HB=1.0;
  q_rate_fcn_MRFSS=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_MRFSS.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_MRFSS_cpue; iyear<=endyr_MRFSS_cpue; iyear++)
      {   if (iyear>styr_MRFSS_cpue & iyear <=2003)
          {//q_rate_fcn_MRFSS(iyear)=(1.0+q_rate)*q_rate_fcn_MRFSS(iyear-1); //compound
             q_rate_fcn_MRFSS(iyear)=(1.0+(iyear-styr_MRFSS_cpue)*q_rate)*q_rate_fcn_MRFSS(styr_MRFSS_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_MRFSS(iyear)=q_rate_fcn_MRFSS(iyear-1);}
      }
  } //end q_rate conditional


  L_mrfss_bias=set_L_mrfss_bias;

  w_L=set_w_L;
  w_D=set_w_D;
  w_lc=set_w_lc;
  w_ac=set_w_ac;
  w_I_RVC=set_w_I_RVC;
  w_I_CVT=set_w_I_CVT;
  w_I_cL=set_w_I_cL;
  w_I_HB=set_w_I_HB;
  w_I_MRFSS=set_w_I_MRFSS;
  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_cO=set_log_avg_F_cO;
  log_avg_F_HB=set_log_avg_F_HB;
  log_avg_F_MRFSS=set_log_avg_F_MRFSS;
  F_init_ratio=set_F_init_ratio;

  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_MRFSS_D=set_log_avg_F_MRFSS_D;
  F_cL_D_ratio=set_F_cL_D_ratio;
  F_HB_D_ratio=set_F_HB_D_ratio;

  log_len_cv=log(set_len_cv);
  log_R0=set_log_R0;

  selpar_L50_RVC=set_selpar_L50_RVC;
  selpar_slope_RVC=set_selpar_slope_RVC;
  selpar_L502_RVC=set_selpar_L502_RVC;
  selpar_slope2_RVC=set_selpar_slope2_RVC;

  selpar_L50_CVT=set_selpar_L50_CVT;
  selpar_slope_CVT=set_selpar_slope_CVT;
  selpar_L502_CVT=set_selpar_L502_CVT;
  selpar_slope2_CVT=set_selpar_slope2_CVT;

  selpar_L50_cL2=set_selpar_L50_cL2;
  selpar_slope_cL2=set_selpar_slope_cL2;
  selpar_L502_cL2=set_selpar_L502_cL2;
  selpar_slope2_cL2=set_selpar_slope2_cL2;
  selpar_L50_cL3=set_selpar_L50_cL3;
  selpar_slope_cL3=set_selpar_slope_cL3;
  selpar_L502_cL3=set_selpar_L502_cL3;
  selpar_slope2_cL3=set_selpar_slope2_cL3;


  selpar_L50_cO2=set_selpar_L50_cO2;
  selpar_L502_cO2=set_selpar_L502_cO2;
  selpar_slope_cO2=set_selpar_slope_cO2;
  selpar_slope2_cO2=set_selpar_slope2_cO2;
  selpar_L50_cO3=set_selpar_L50_cO3;

  selpar_L50_HB1=set_selpar_L50_HB1;
  selpar_slope_HB1=set_selpar_slope_HB1;
  selpar_L502_HB1=set_selpar_L502_HB1;
  selpar_slope2_HB1=set_selpar_slope2_HB1;
  selpar_L50_HB2=set_selpar_L50_HB2;
  selpar_slope_HB2=set_selpar_slope_HB2;
  selpar_L502_HB2=set_selpar_L502_HB2;
  selpar_slope2_HB2=set_selpar_slope2_HB2;
  selpar_L50_HB3=set_selpar_L50_HB3;
  selpar_slope_HB3=set_selpar_slope_HB3;
  selpar_L502_HB3=set_selpar_L502_HB3;
  selpar_slope2_HB3=set_selpar_slope2_HB3;


//  sel_HB_D_2=set_sel_HB_D_2;
//  sel_HB_D_3=set_sel_HB_D_3;
  selpar_Age1_HB_D3=set_selpar_Age1_HB_D3;
//  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_MRFSS3=set_selpar_L50_MRFSS3;
  selpar_slope_MRFSS3=set_selpar_slope_MRFSS3;
  selpar_L502_MRFSS3=set_selpar_L502_MRFSS3;
  selpar_slope2_MRFSS3=set_selpar_slope2_MRFSS3;


 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;
 maturity_m=maturity_m_obs;
 prop_f=prop_f_obs;

 p_lenc_cL2=set_p_lenc_cL2;
 p_lenc_cL3=set_p_lenc_cL3;
 p_lenc_cO2=set_p_lenc_cO2;
 p_lenc_cO3=set_p_lenc_cO3;
 p_lenc_HB2=set_p_lenc_HB2;
 p_lenc_HB3=set_p_lenc_HB3;
 p_lenc_MRFSS2=set_p_lenc_MRFSS2;
 p_lenc_MRFSS3=set_p_lenc_MRFSS3;

 p_lenc_cL_D2=set_p_lenc_cL_D2;
 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_MRFSS_D2=set_p_lenc_MRFSS_D2;
 p_lenc_MRFSS_D3=set_p_lenc_MRFSS_D3;

////Fill in maturity matrix for calculations for styr to styr
//    for(iyear=styr; iyear<=styr-1; iyear++)
//      {
//         maturity_f(iyear)=maturity_f_obs;
//         maturity_m(iyear)=maturity_m_obs;
//         prop_m(iyear)=prop_m_obs(styr);
//         prop_f(iyear)=1.0-prop_m_obs(styr);
//      }
//    for (iyear=styr;iyear<=endyr;iyear++)
//      {
//         maturity_f(iyear)=maturity_f_obs;
//         maturity_m(iyear)=maturity_m_obs;
//         prop_m(iyear)=prop_m_obs(iyear);
//         prop_f(iyear)=1.0-prop_m_obs(iyear);
//      }

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

      nsamp_RVC_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nsamp_CVT_lenc_allyr=missing;
      nsamp_cL_lenc_allyr=missing;
      nsamp_cO_lenc_allyr=missing;
      nsamp_HB_lenc_allyr=missing;
      nsamp_HB_D_lenc_allyr=missing;
      nsamp_MRFSS_lenc_allyr=missing;
      nsamp_CVT_agec_allyr=missing;
      nsamp_cL_agec_allyr=missing;
      nsamp_HB_agec_allyr=missing;
      nsamp_MRFSS_agec_allyr=missing;

      neff_RVC_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      neff_CVT_lenc_allyr=missing;
      neff_cL_lenc_allyr=missing;
      neff_cO_lenc_allyr=missing;
      neff_HB_lenc_allyr=missing;
      neff_HB_D_lenc_allyr=missing;
      neff_MRFSS_lenc_allyr=missing;
      neff_CVT_agec_allyr=missing;
      neff_cL_agec_allyr=missing;
      neff_HB_agec_allyr=missing;
      neff_MRFSS_agec_allyr=missing;

      for (iyear=styr_RVC_lenc; iyear<=endyr_RVC_lenc; iyear++)
         {
           if (nsamp_RVC_lenc(iyear)>=minSS_RVC_lenc)
           {
             nsamp_RVC_lenc_allyr(iyear)=nsamp_RVC_lenc(iyear);
             neff_RVC_lenc_allyr(iyear)=neff_RVC_lenc(iyear);
           }
         }
      for (iyear=styr_CVT_lenc; iyear<=endyr_CVT_lenc; iyear++)
         {
           if (nsamp_CVT_lenc(iyear)>=minSS_CVT_lenc)
           {
             nsamp_CVT_lenc_allyr(iyear)=nsamp_CVT_lenc(iyear);
             neff_CVT_lenc_allyr(iyear)=neff_CVT_lenc(iyear);
           }
         }
      for (iyear=styr_cL_lenc; iyear<=endyr_cL_lenc; iyear++)
         {
           if (nsamp_cL_lenc(iyear)>=minSS_cL_lenc)
           {
             nsamp_cL_lenc_allyr(iyear)=nsamp_cL_lenc(iyear);
             neff_cL_lenc_allyr(iyear)=neff_cL_lenc(iyear);
           }
         }
      for (iyear=1; iyear<=nyr_cO_lenc; iyear++)
         {
           if (nsamp_cO_lenc(iyear)>=minSS_cO_lenc)
           {
             nsamp_cO_lenc_allyr(yrs_cO_lenc(iyear))=nsamp_cO_lenc(iyear);
             neff_cO_lenc_allyr(yrs_cO_lenc(iyear))=neff_cO_lenc(iyear);
           }
         }
      for (iyear=styr_HB_lenc; iyear<=endyr_HB_lenc; iyear++)
         {
           if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
           {
             nsamp_HB_lenc_allyr(iyear)=nsamp_HB_lenc(iyear);
             neff_HB_lenc_allyr(iyear)=neff_HB_lenc(iyear);
           }
         }
      for (iyear=styr_HB_D_lenc; iyear<=endyr_HB_D_lenc; iyear++)
         {
           if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
           {
             nsamp_HB_D_lenc_allyr(iyear)=nsamp_HB_D_lenc(iyear);
             neff_HB_D_lenc_allyr(iyear)=neff_HB_D_lenc(iyear);
           }
         }
      for (iyear=styr_MRFSS_lenc; iyear<=endyr_MRFSS_lenc; iyear++)
         {
           if (nsamp_MRFSS_lenc(iyear)>=minSS_MRFSS_lenc)
           {
             nsamp_MRFSS_lenc_allyr(iyear)=nsamp_MRFSS_lenc(iyear);
             neff_MRFSS_lenc_allyr(iyear)=neff_MRFSS_lenc(iyear);
           }
         }
      for (iyear=styr_CVT_agec; iyear<=endyr_CVT_agec; iyear++)
         {
           if (nsamp_CVT_agec(iyear)>=minSS_CVT_agec)
           {
             nsamp_CVT_agec_allyr(iyear)=nsamp_CVT_agec(iyear);
             neff_CVT_agec_allyr(iyear)=neff_CVT_agec(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_HB_agec; iyear++)
         {
           if (nsamp_HB_agec(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=styr_MRFSS_agec; iyear<=endyr_MRFSS_agec; iyear++)
         {
           if (nsamp_MRFSS_agec(iyear)>=minSS_MRFSS_agec)
           {
             nsamp_MRFSS_agec_allyr(iyear)=nsamp_MRFSS_agec(iyear);
             neff_MRFSS_agec_allyr(iyear)=neff_MRFSS_agec(iyear);
           }
         }

//fill in Fs for msy and per-recruit analyses
  F_msy(1)=0.0;
  for (int 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_cO.initialize();
  L_cO_num.initialize();
  F_HB.initialize();
  L_HB_num.initialize();
  F_MRFSS.initialize();
  L_MRFSS_num.initialize();

  F_cL_out.initialize();
  F_cO_out.initialize();
  F_HB_out.initialize();
  F_MRFSS_out.initialize();

  F_cL_D_out.initialize();
  F_HB_D_out.initialize();
  F_MRFSS_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_MRFSS_D.initialize();
  D_MRFSS_num.initialize();
  pred_MRFSS_D_klb.initialize();

  L_total_knum_yr.initialize();
  L_total_klb_yr.initialize();

  sel_cL_D.initialize();
  sel_HB_D.initialize();
  sel_MRFSS_D.initialize();

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


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


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

 R0=mfexp(log_R0);
 //cout<<"start"<<endl;
 get_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_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

FUNCTION get_reprod

   //product of stuff going into reproductive capacity calcs
   reprod=elem_prod((elem_prod(prop_f,maturity_f)+elem_prod((1.0-prop_f),maturity_m)),wgt_mt);

FUNCTION get_length_at_age_dist
  //compute matrix of length at age, based on the normal distribution
  for (iage=1;iage<=nages;iage++)
  {
    //len_cv(iage)=mfexp(log_len_cv+log_len_cv_dev(iage));
    len_cv(iage)=mfexp(log_len_cv);
    for (ilen=1;ilen<=nlenbins;ilen++)
    {
      lenprob(iage,ilen)=(mfexp(-(square(lenbins(ilen)-meanlen_TL(iage))/
      (2.*square(len_cv(iage)*meanlen_TL(iage)))))/(sqrt2pi*len_cv(iage)*meanlen_TL(iage)));
    }
    for (ilen=1;ilen<=nlenbins_plus;ilen++)
    {
      lenprob_plus(iage,ilen)=(mfexp(-(square(lenbins_plus(ilen)-meanlen_TL(iage))/
      (2.*square(len_cv(iage)*meanlen_TL(iage)))))/(sqrt2pi*len_cv(iage)*meanlen_TL(iage)));
    }

    lenprob(iage)(nlenbins)=lenprob(iage)(nlenbins)+sum(lenprob_plus(iage)); //add mass to plus group
    lenprob(iage)/=sum(lenprob(iage)); //standardize to approximate integration and to account for truncated normal (i.e., no sizes<smallest)
  }

  //fishery specific length probs, assumed normal prior to size limits

  lenprob_cL1=lenprob;
  lenprob_cL2=lenprob; //values may be adjusted later based on size limit
  lenprob_cL3=lenprob; //values may be adjusted based on size limit

  lenprob_cO1=lenprob;
  lenprob_cO2=lenprob; //values may be adjusted based on size limit
  lenprob_cO3=lenprob; //values may be adjusted based on size limit

  lenprob_HB1=lenprob;
  lenprob_HB2=lenprob; //values may be adjusted based on size limit
  lenprob_HB3=lenprob; //values may be adjusted based on size limit
  lenprob_MRFSS1=lenprob;
  lenprob_MRFSS2=lenprob; //values may be adjusted based on size limit
  lenprob_MRFSS3=lenprob; //values may be adjusted based on size limit

  lenprob_cL_D3=lenprob; //values may be adjusted based on size limit
  lenprob_HB_D2=lenprob; //values may be adjusted based on size limit
  lenprob_HB_D3=lenprob; //values may be adjusted based on size limit
  lenprob_MRFSS_D2=lenprob; //values may be adjusted based on size limit
  lenprob_MRFSS_D3=lenprob; //values may be adjusted based on size limit


  for (iage=1;iage<=nages;iage++)
  {
    for (ilen=1;ilen<=nlenbins;ilen++)
    {
        if (lenbins(ilen) < limit_12in)
        {
          lenprob_cL2(iage,ilen)=p_lenc_cL2*lenprob(iage,ilen);
          lenprob_cO2(iage,ilen)=p_lenc_cO2*lenprob(iage,ilen);
          lenprob_HB2(iage,ilen)=p_lenc_HB2*lenprob(iage,ilen);
          lenprob_MRFSS2(iage,ilen)=p_lenc_MRFSS2*lenprob(iage,ilen);

        }
        if (lenbins(ilen) < limit_20in)
        {
          lenprob_cL3(iage,ilen)=p_lenc_cL3*lenprob(iage,ilen);
          lenprob_cO3(iage,ilen)=p_lenc_cO3*lenprob(iage,ilen);
          lenprob_HB3(iage,ilen)=p_lenc_HB3*lenprob(iage,ilen);
          lenprob_MRFSS3(iage,ilen)=p_lenc_MRFSS3*lenprob(iage,ilen);
         }


        if (lenbins(ilen) > limit_disc) //block two
        { lenprob_cL_D2(iage,ilen)=p_lenc_cL_D2*lenprob(iage,ilen);
          lenprob_HB_D2(iage,ilen)=p_lenc_HB_D2*lenprob(iage,ilen);
          lenprob_MRFSS_D2(iage,ilen)=p_lenc_MRFSS_D2*lenprob(iage,ilen);
        }


        if (lenbins(ilen) > limit_20in) //block three
        { lenprob_cL_D3(iage,ilen)=p_lenc_cL_D3*lenprob(iage,ilen);
          lenprob_HB_D3(iage,ilen)=p_lenc_HB_D3*lenprob(iage,ilen);
          lenprob_MRFSS_D3(iage,ilen)=p_lenc_MRFSS_D3*lenprob(iage,ilen);
        }

    }  //end ilen loop

    if (iage>=3)  //compute prior to standardizing
        {vecprob_HB_D2(iage)=sum(lenprob_HB_D2(iage));
         vecprob_HB_D3(iage)=sum(lenprob_HB_D3(iage));}

    lenprob_cL2(iage)/=sum(lenprob_cL2(iage));       //standardize
    lenprob_cL3(iage)/=sum(lenprob_cL3(iage));       //standardize
    lenprob_cO2(iage)/=sum(lenprob_cO2(iage));       //standardize
    lenprob_cO3(iage)/=sum(lenprob_cO3(iage));       //standardize
    lenprob_HB2(iage)/=sum(lenprob_HB2(iage));       //standardize
    lenprob_MRFSS2(iage)/=sum(lenprob_MRFSS2(iage)); //standardize
    lenprob_HB3(iage)/=sum(lenprob_HB3(iage));       //standardize
    lenprob_MRFSS3(iage)/=sum(lenprob_MRFSS3(iage)); //standardize

    lenprob_cL_D3(iage)/=sum(lenprob_cL_D3(iage)); //standardize
    lenprob_HB_D2(iage)/=sum(lenprob_HB_D2(iage)); //standardize
    lenprob_HB_D3(iage)/=sum(lenprob_HB_D3(iage)); //standardize
    lenprob_MRFSS_D2(iage)/=sum(lenprob_MRFSS_D2(iage)); //standardize
    lenprob_MRFSS_D3(iage)/=sum(lenprob_MRFSS_D3(iage)); //standardize

  } //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_cO_mm(iyear)=meanlen_TL;
    wgt_cO_klb(iyear)=wgt_klb;
    len_HB_mm(iyear)=meanlen_TL;
    wgt_HB_klb(iyear)=wgt_klb;
    len_MRFSS_mm(iyear)=meanlen_TL;
    wgt_MRFSS_klb(iyear)=wgt_klb;

    //HB and rec discards assumed same size as period one, consistent with selectivity assumptions
    for (iage=1;iage<=nages; iage++)
    {
      len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D2(iage),lenbins));
    }
      wgt_HB_D_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_D_mm(iyear),wgtpar_b);
  } // end iyear loop


  for (iyear=(endyr_period1+1); iyear<=endyr_period2; iyear++)
  {
    for (iage=1;iage<=nages; iage++)
    {
    len_cL_mm(iyear,iage)=sum(elem_prod(lenprob_cL2(iage),lenbins));
    len_cO_mm(iyear,iage)=sum(elem_prod(lenprob_cO2(iage),lenbins));
    len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB2(iage),lenbins));
    len_MRFSS_mm(iyear,iage)=sum(elem_prod(lenprob_MRFSS2(iage),lenbins));
//    len_cL_D_mm(iyear,iage)=sum(elem_prod(lenprob_cL_D2(iage),lenbins));
    len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D2(iage),lenbins));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    wgt_cO_klb(iyear)=g2klb*wgtpar_a*pow(len_cO_mm(iyear),wgtpar_b);
    wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_MRFSS_klb(iyear)=g2klb*wgtpar_a*pow(len_MRFSS_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(iage),lenbins));
    len_cO_mm(iyear,iage)=sum(elem_prod(lenprob_cO3(iage),lenbins));
    len_HB_mm(iyear,iage)=sum(elem_prod(lenprob_HB3(iage),lenbins));
    len_MRFSS_mm(iyear,iage)=sum(elem_prod(lenprob_MRFSS3(iage),lenbins));
    len_cL_D_mm(iyear,iage)=sum(elem_prod(lenprob_cL_D3(iage),lenbins));
    len_HB_D_mm(iyear,iage)=sum(elem_prod(lenprob_HB_D3(iage),lenbins));
    }
    wgt_cL_klb(iyear)=g2klb*wgtpar_a*pow(len_cL_mm(iyear),wgtpar_b);
    wgt_cO_klb(iyear)=g2klb*wgtpar_a*pow(len_cO_mm(iyear),wgtpar_b);
    wgt_HB_klb(iyear)=g2klb*wgtpar_a*pow(len_HB_mm(iyear),wgtpar_b);
    wgt_MRFSS_klb(iyear)=g2klb*wgtpar_a*pow(len_MRFSS_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);
  }

  //MRFSS_D assumed same as HB_D
  len_MRFSS_D_mm=len_HB_D_mm;
  wgt_MRFSS_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

  selpar_slope_cO2=selpar_slope_CVT;
  selpar_slope2_cO2=selpar_slope2_CVT;
  selpar_L502_cO2=selpar_L502_CVT;
  selpar_slope_cO3=selpar_slope_CVT;
  selpar_slope2_cO3=selpar_slope2_CVT;
  selpar_L502_cO3=selpar_L502_CVT;


  for (iage=1; iage<=nages; iage++)
  {
     //sel_RVC_vec(iage)=(1./(1.+mfexp(-1.*selpar_slope_RVC*(double(agebins(iage))-
     //                  selpar_L50_RVC))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_RVC*
     //                  (double(agebins(iage))-(selpar_L50_RVC+selpar_L502_RVC)))))); //double logistic
     sel_RVC_vec(iage)=1./(1.+mfexp(selpar_slope_RVC*(double(agebins(iage))-selpar_L50_RVC))); //declining logistic


     sel_CVT_vec(iage)=(1./(1.+mfexp(-1.*selpar_slope_CVT*(double(agebins(iage))-
                       selpar_L50_CVT))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_CVT*
                       (double(agebins(iage))-(selpar_L50_CVT+selpar_L502_CVT)))))); //double logistic

//     //sel_cL_1(iage)=1./(1.+mfexp(-1.*selpar_slope_cL1*(double(agebins(iage))-selpar_L50_cL1))); //logistic
//     sel_cL_1(iage)=(1./(1.+mfexp(-1.*selpar_slope_cL1*(double(agebins(iage))-
//                       selpar_L50_cL1))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cL1*
//                       (double(agebins(iage))-(selpar_L50_cL1+selpar_L502_cL1)))))); //double logistic

     sel_cL_2(iage)=1./(1.+mfexp(-1.*selpar_slope_cL2*(double(agebins(iage))-selpar_L50_cL2))); //logistic
//     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
//
     sel_cL_3(iage)=1./(1.+mfexp(-1.*selpar_slope_cL3*(double(agebins(iage))-selpar_L50_cL3))); //logistic
//     sel_cL_3(iage)=(1./(1.+mfexp(-1.*selpar_slope_cL3*(double(agebins(iage))-
//                       selpar_L50_cL3))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cL3*
//                       (double(agebins(iage))-(selpar_L50_cL3+selpar_L502_cL3)))))); //double logistic

//     sel_cO_1(iage)=(1./(1.+mfexp(-1.*selpar_slope_cO1*(double(agebins(iage))-
//                       selpar_L50_cO1))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cO1*
//                       (double(agebins(iage))-(selpar_L50_cO1+selpar_L502_cO1)))))); //double logistic
     sel_cO_2(iage)=(1./(1.+mfexp(-1.*selpar_slope_cO2*(double(agebins(iage))-
                       selpar_L50_cO2))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cO2*
                       (double(agebins(iage))-(selpar_L50_cO2+selpar_L502_cO2)))))); //double logistic
     sel_cO_3(iage)=(1./(1.+mfexp(-1.*selpar_slope_cO3*(double(agebins(iage))-
                       selpar_L50_cO3))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_cO3*
                       (double(agebins(iage))-(selpar_L50_cO3+selpar_L502_cO3)))))); //double logistic


     sel_HB_1(iage)=1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))-selpar_L50_HB1))); //logistic
     sel_HB_1(iage)=(1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))-
                       selpar_L50_HB1))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_HB1*
                       (double(agebins(iage))-(selpar_L50_HB1+selpar_L502_HB1)))))); //double logistic
     //sel_HB_2(iage)=1./(1.+mfexp(-1.*selpar_slope_HB2*(double(agebins(iage))-selpar_L50_HB2))); //logistic
     sel_HB_2(iage)=(1./(1.+mfexp(-1.*selpar_slope_HB2*(double(agebins(iage))-
                       selpar_L50_HB2))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_HB2*
                       (double(agebins(iage))-(selpar_L50_HB2+selpar_L502_HB2)))))); //double logistic
     //sel_HB_3(iage)=1./(1.+mfexp(-1.*selpar_slope_HB3*(double(agebins(iage))-selpar_L50_HB3))); //logistic
     sel_HB_3(iage)=(1./(1.+mfexp(-1.*selpar_slope_HB3*(double(agebins(iage))-
                       selpar_L50_HB3))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_HB3*
                       (double(agebins(iage))-(selpar_L50_HB3+selpar_L502_HB3)))))); //double logistic
     //sel_MRFSS_3(iage)=1./(1.+mfexp(-1.*selpar_slope_MRFSS3*(double(agebins(iage))-selpar_L50_MRFSS3))); //logistic
     sel_MRFSS_3(iage)=(1./(1.+mfexp(-1.*selpar_slope_MRFSS3*(double(agebins(iage))-
                       selpar_L50_MRFSS3))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_MRFSS3*
                       (double(agebins(iage))-(selpar_L50_MRFSS3+selpar_L502_MRFSS3)))))); //double logistic

  }
  sel_RVC_vec=sel_RVC_vec/max(sel_RVC_vec); //re-normalize double logistic
  sel_CVT_vec=sel_CVT_vec/max(sel_CVT_vec); //re-normalize double logistic

  //sel_cL_1=sel_cL_1/max(sel_cL_1);    //re-normalize double logistic
  sel_cL_2=sel_cL_2/max(sel_cL_2);    //re-normalize double logistic
  sel_cL_3=sel_cL_3/max(sel_cL_3);    //re-normalize double logistic

//  sel_cO_1=sel_cO_1/max(sel_cO_1);    //re-normalize double logistic
  sel_cO_2=sel_cO_2/max(sel_cO_2);    //re-normalize double logistic
  sel_cO_3=sel_cO_3/max(sel_cO_3);    //re-normalize double logistic

  sel_HB_1=sel_HB_1/max(sel_HB_1); //re-normalize double logistic
  sel_HB_2=sel_HB_2/max(sel_HB_2); //re-normalize double logistic
  sel_HB_3=sel_HB_3/max(sel_HB_3); //re-normalize double logistic

//  //sel_MRFSS_1=sel_HB_1;
//  //sel_MRFSS_2=sel_HB_2;
  sel_MRFSS_3=sel_MRFSS_3/max(sel_MRFSS_3); //re-normalize double logistic


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

  for (iyear=styr; iyear<=endyr; iyear++)
  {  //time-invariant selectivities
     sel_RVC(iyear)=sel_RVC_vec;
     sel_CVT(iyear)=sel_CVT_vec;

  }
  //Period 1:
  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
     sel_cL(iyear)=sel_cL_2; //early commercial sel same as in subsequent period
     sel_cO(iyear)=sel_cO_2; //early commercial sel same as in subsequent period
     sel_HB(iyear)=sel_HB_1;
     sel_MRFSS(iyear)=sel_HB_1; //MRFSS same as HB.
  }

  //Period 2:
  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  {
     sel_cL(iyear)=sel_cL_2;
     sel_cO(iyear)=sel_cO_2;
     sel_HB(iyear)=sel_HB_2;
     sel_MRFSS(iyear)=sel_HB_2; //MRFSS same as HB
  }

  //Period 3
  for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
  {
     sel_cL(iyear)=sel_cL_3;
     sel_cO(iyear)=sel_cO_3;
     sel_HB(iyear)=sel_HB_3;
     sel_MRFSS(iyear)=sel_MRFSS_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_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_HB_D_2=sel_HB_D_2/max(sel_HB_D_2); //re-normalize double logistic
  sel_HB_D_3=sel_HB_D_3/max(sel_HB_D_3); //re-normalize double logistic

  //Period 1: All fleets same as HB and same as in period 2
  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {sel_HB_D(iyear)=sel_HB_D_2;}

  //Period 2: all fleets same as HB
  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;}
  //Commercial and Rec discard selectivities same as headboat
  sel_cL_D=sel_HB_D;
  sel_MRFSS_D=sel_HB_D;

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  ////initialization F is avg of first 3 yrs of observed landings
  log_F_dev_init_cL=sum(log_F_dev_cL(styr_cL_L,(styr_cL_L+2)))/3.0;
  log_F_dev_init_cO=sum(log_F_dev_cO(styr_cO_L,(styr_cO_L+2)))/3.0;
  log_F_init_HB=sum(log_F_dev_HB(styr_HB_L,(styr_HB_L+2)))/3.0;
  log_F_dev_init_MRFSS=sum(log_F_dev_MRFSS(styr_MRFSS_L,(styr_MRFSS_L+2)))/3.0;


  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_cO_L & iyear<=endyr_cO_L)
        {F_cO_out(iyear)=mfexp(log_avg_F_cO+log_F_dev_cO(iyear));}
    if (iyear<styr_cO_L)
      {F_cO_out(iyear)=mfexp(log_avg_F_cO+log_F_dev_init_cO);}
    F_cO(iyear)=sel_cO(iyear)*F_cO_out(iyear);
    Fsum(iyear)+=F_cO_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_MRFSS_L & iyear<=endyr_MRFSS_L)
    {F_MRFSS_out(iyear)=mfexp(log_avg_F_MRFSS+log_F_dev_MRFSS(iyear));}
    if (iyear<styr_MRFSS_L)
      {F_MRFSS_out(iyear)=mfexp(log_avg_F_MRFSS+log_F_dev_init_MRFSS);}
    F_MRFSS(iyear)=sel_MRFSS(iyear)*F_MRFSS_out(iyear);
    Fsum(iyear)+=F_MRFSS_out(iyear);

    //discards-------------
    log_F_avgdev_cL_D=sum(log_F_dev_cL_D(styr_cL_D,endyr_cL_D))/(endyr_cL_D-styr_cL_D+1.0);
    log_F_avgdev_HB_D=sum(log_F_dev_HB_D(styr_HB_D,endyr_HB_D))/(endyr_HB_D-styr_HB_D+1.0);

    if(iyear>=styr_cL_D)
      {F_cL_D_out(iyear)=mfexp(log_avg_F_cL_D+log_F_dev_cL_D(iyear));}
    if(iyear>endyr_period1 & iyear<styr_cL_D)
      {F_cL_D_out(iyear)=F_cL_D_ratio*mfexp(log_avg_F_cL_D+log_F_avgdev_cL_D);}
    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));}
    if(iyear>endyr_period1 & iyear<=endyr_period2)
      {F_HB_D_out(iyear)=F_HB_D_ratio*mfexp(log_avg_F_HB_D+log_F_avgdev_HB_D);}
    if(iyear>endyr_period2 & iyear<styr_HB_D)
      {F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_avgdev_HB_D);}
    if(iyear>endyr_HB_D)
      {F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_avgdev_HB_D);}
    F_HB_D(iyear)=sel_HB_D(iyear)*F_HB_D_out(iyear);
    Fsum(iyear)+=F_HB_D_out(iyear);

    if(iyear>=styr_MRFSS_D)
    {
      F_MRFSS_D_out(iyear)=mfexp(log_avg_F_MRFSS_D+log_F_dev_MRFSS_D(iyear));
      F_MRFSS_D(iyear)=sel_MRFSS_D(iyear)*F_MRFSS_D_out(iyear);
      Fsum(iyear)+=F_MRFSS_D_out(iyear);
    }

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

    F(iyear)+=F_cL_D(iyear);
    F(iyear)+=F_HB_D(iyear);
    F(iyear)+=F_MRFSS_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
  else {BiasCor=set_BiasCor;}


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

  F_initial=sel_cL(styr)*mfexp(log_avg_F_cL+log_F_dev_init_cL)+
           sel_cO(styr)*mfexp(log_avg_F_cO+log_F_dev_init_cO)+
           sel_HB(styr)*mfexp(log_avg_F_HB+log_F_init_HB)+
           sel_MRFSS(styr)*mfexp(log_avg_F_MRFSS+log_F_dev_init_MRFSS);
  Z_initial=M+F_init_ratio*F_initial;


//Initial equilibrium age structure
  N_spr_initial(1)=1.0*mfexp(-1.0*Z_initial(1)*spawn_time_frac); //at peak spawning time;
  for (iage=2; iage<=nages; iage++)
    {
      N_spr_initial(iage)=N_spr_initial(iage-1)*
                   mfexp(-1.0*(Z_initial(iage-1)*(1.0-spawn_time_frac) + Z_initial(iage)*spawn_time_frac));
    }
  N_spr_initial(nages)=N_spr_initial(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group
```

```
//    N_spr_F_init_mdyr(1,(nages-1))=elem_prod(N_spr_initial(1,(nages-1)),
//                                   mfexp((-1.*(M(nages-1)+ F_initial))/2.0));

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

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

  if(R1<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)*(1.0-spawn_time_frac) + Z_initial(iage)*spawn_time_frac));
  }
  //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));
  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)(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));
        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));
        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_cO_num(iyear,iage)=N(iyear,iage)*F_cO(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_MRFSS_num(iyear,iage)=N(iyear,iage)*F_MRFSS(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }

    pred_cL_L_knum(iyear)=sum(L_cL_num(iyear))/1000.0;
    pred_cO_L_knum(iyear)=sum(L_cO_num(iyear))/1000.0;
    pred_HB_L_knum(iyear)=sum(L_HB_num(iyear))/1000.0;
    pred_MRFSS_L_knum(iyear)=sum(L_MRFSS_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_cO_klb(iyear)=elem_prod(L_cO_num(iyear),wgt_cO_klb(iyear));     //in 1000 lb
    L_HB_klb(iyear)=elem_prod(L_HB_num(iyear),wgt_HB_klb(iyear));            //in 1000 lb
    L_MRFSS_klb(iyear)=elem_prod(L_MRFSS_num(iyear),wgt_MRFSS_klb(iyear));   //in 1000 lb

    pred_cL_L_klb(iyear)=sum(L_cL_klb(iyear));
    pred_cO_L_klb(iyear)=sum(L_cO_klb(iyear));
    pred_HB_L_klb(iyear)=sum(L_HB_klb(iyear));
    pred_MRFSS_L_klb(iyear)=sum(L_MRFSS_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_MRFSS_num(iyear,iage)=N(iyear,iage)*F_MRFSS_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_MRFSS_D_knum(iyear)=sum(D_MRFSS_num(iyear))/1000.0;             //pred annual dead discards in 1000s (for matching data)
    pred_MRFSS_D_klb(iyear)=sum(elem_prod(D_MRFSS_num(iyear),wgt_MRFSS_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_MRFSS_cpue; iyear<=endyr_MRFSS_cpue; iyear++)
      {   if (iyear>styr_MRFSS_cpue & iyear <=2003)
          {//q_rate_fcn_MRFSS(iyear)=(1.0+q_rate)*q_rate_fcn_MRFSS(iyear-1); //compound
             q_rate_fcn_MRFSS(iyear)=(1.0+(iyear-styr_MRFSS_cpue)*q_rate)*q_rate_fcn_MRFSS(styr_MRFSS_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_MRFSS(iyear)=q_rate_fcn_MRFSS(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------------------------


 //RVC survey
  for (iyear=styr_RVC_cpue; iyear<=endyr_RVC_cpue; iyear++)
  {   //index in number units
      N_RVC(iyear)=elem_prod(N_mdyr(iyear),sel_RVC(iyear));
      pred_RVC_cpue(iyear)=mfexp(log_q_RVC)*sum(N_RVC(iyear));
  }

 //MARMAP chevron trap
  for (iyear=styr_CVT_cpue; iyear<=endyr_CVT_cpue; iyear++)
  {   //index in number units
      N_CVT(iyear)=elem_prod(N_mdyr(iyear),sel_CVT(iyear));
      pred_CVT_cpue(iyear)=mfexp(log_q_CVT)*sum(N_CVT(iyear));
  }

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

 //Headboat cpue
  q_HB(styr_HB_cpue)=mfexp(log_q_HB);
  for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
  {   //index in 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));}
  }
  //MRFSS cpue
  sel_MRFSS_cpue.initialize(); //approximate selectivity; includes discards (live + dead)
  q_MRFSS(styr_MRFSS_cpue)=mfexp(log_q_MRFSS);
  for (iyear=styr_MRFSS_cpue; iyear<=endyr_MRFSS_cpue; iyear++)
  {   //index in number units
      sel_MRFSS_cpue(iyear)=F_MRFSS(iyear)+F_MRFSS_D(iyear)/Dmort_MRFSS;
      sel_MRFSS_cpue(iyear)/=max(sel_MRFSS_cpue(iyear));
      N_MRFSS(iyear)=elem_prod(N_mdyr(iyear),sel_MRFSS_cpue(iyear));
      pred_MRFSS_cpue(iyear)=q_MRFSS(iyear)*q_rate_fcn_MRFSS(iyear)*q_DD_fcn(iyear)*sum(N_MRFSS(iyear));
      if (iyear<endyr_MRFSS_cpue){q_MRFSS(iyear+1)=q_MRFSS(iyear)*mfexp(q_RW_log_dev_MRFSS(iyear));}
  }


FUNCTION get_length_comps
 //Fishery independent
  for (iyear=styr_RVC_lenc;iyear<=endyr_RVC_lenc;iyear++)
  {
    pred_RVC_lenc(iyear)=(N_RVC(iyear)*lenprob)/sum(N_RVC(iyear));
  }

  for (iyear=styr_CVT_lenc;iyear<=endyr_CVT_lenc;iyear++)
  {
    pred_CVT_lenc(iyear)=(N_CVT(iyear)*lenprob)/sum(N_CVT(iyear));
  }

 //Commercial
  for (iyear=styr_cL_lenc; iyear<=endyr_period2; iyear++)
  {
     pred_cL_lenc(iyear)=(L_cL_num(iyear)*lenprob_cL2)/sum(L_cL_num(iyear));
  }

  for (iyear=(endyr_period2+1);iyear<=endyr_cL_lenc;iyear++)
  {
      pred_cL_lenc(iyear)=(L_cL_num(iyear)*lenprob_cL3)/sum(L_cL_num(iyear));
  }

  for (iyear=1;iyear<=nyr_cO_lenc;iyear++) //all yrs within period 2
  {
         pred_cO_lenc(iyear)=(L_cO_num(yrs_cO_lenc(iyear))*lenprob_cO2)
                                 /sum(L_cO_num(yrs_cO_lenc(iyear)));
  }


 //Headboat
  for (iyear=styr_HB_lenc;iyear<=endyr_period1;iyear++)
  {
    pred_HB_lenc(iyear)=(L_HB_num(iyear)*lenprob_HB1)/sum(L_HB_num(iyear));
  }
  for (iyear=(endyr_period1+1);iyear<=endyr_period2;iyear++)
  {
    pred_HB_lenc(iyear)=(L_HB_num(iyear)*lenprob_HB2)/sum(L_HB_num(iyear));
  }
  for (iyear=(endyr_period2+1);iyear<=endyr_HB_lenc;iyear++)
  {
    pred_HB_lenc(iyear)=(L_HB_num(iyear)*lenprob_HB3)/sum(L_HB_num(iyear));
  }

  for (iyear=styr_HB_D_lenc;iyear<=endyr_HB_D_lenc;iyear++)
  {
    pred_HB_D_lenc(iyear)=(D_HB_num(iyear)*lenprob_HB_D3)/sum(D_HB_num(iyear));
  }



 //MRFSS All in period 3
  for (iyear=styr_MRFSS_lenc;iyear<=endyr_MRFSS_lenc;iyear++)
  {
    pred_MRFSS_lenc(iyear)=(L_MRFSS_num(iyear)*lenprob_MRFSS3)/sum(L_MRFSS_num(iyear));
  }


FUNCTION get_age_comps

 //MARMAP CVT
   for (iyear=styr_CVT_agec;iyear<=endyr_CVT_agec;iyear++)
  {
    ErrorFree_CVT_agec(iyear)=N_CVT(iyear)/sum(N_CVT(iyear));
    pred_CVT_agec(iyear)=age_error*ErrorFree_CVT_agec(iyear);
  }

 //Commercial
  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);
  }

 //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);
  }

 //MRFSS
  for (iyear=styr_MRFSS_agec;iyear<=endyr_MRFSS_agec;iyear++)
  {
    ErrorFree_MRFSS_agec(iyear)=L_MRFSS_num(iyear)/sum(L_MRFSS_num(iyear));
    pred_MRFSS_agec(iyear)=age_error*ErrorFree_MRFSS_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_cO+
        sum(log_F_dev_cO((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_MRFSS+
        sum(log_F_dev_MRFSS((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_MRFSS_D+
        sum(log_F_dev_MRFSS_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_cO_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cO+
        sum(log_F_dev_cO((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_MRFSS_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_MRFSS+
        sum(log_F_dev_MRFSS((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_MRFSS_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_MRFSS_D+
        sum(log_F_dev_MRFSS_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_cO=sum(log_F_dev_cO((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_MRFSS=sum(log_F_dev_MRFSS((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_MRFSS_D=sum(log_F_dev_MRFSS_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_cO(endyr)*mfexp(log_avg_F_cO+log_F_dev_end_cO)+
          sel_HB(endyr)*mfexp(log_avg_F_HB+log_F_dev_end_HB)+
          sel_MRFSS(endyr)*mfexp(log_avg_F_MRFSS+log_F_dev_end_MRFSS);

  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_MRFSS_D(endyr)*mfexp(log_avg_F_MRFSS_D+log_F_dev_end_MRFSS_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_cO_prop+F_HB_prop+F_MRFSS_prop;
  wgt_wgted_L_klb=F_cL_prop/wgt_wgted_L_denom*wgt_cL_klb(endyr)+
              F_cO_prop/wgt_wgted_L_denom*wgt_cO_klb(endyr)+
              F_HB_prop/wgt_wgted_L_denom*wgt_HB_klb(endyr)+
              F_MRFSS_prop/wgt_wgted_L_denom*wgt_MRFSS_klb(endyr);

  wgt_wgted_D_denom=F_cL_D_prop+F_HB_D_prop+F_MRFSS_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_MRFSS_D_prop/wgt_wgted_D_denom*wgt_MRFSS_D_klb(endyr);

FUNCTION get_msy

  //compute values as functions of F
  for(int 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

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

  L_total_num=(L_HB_num+L_MRFSS_num+L_cL_num+L_cO_num); //catch in number fish
  L_total_klb=L_HB_klb+L_MRFSS_klb+L_cL_klb+L_cO_klb;   //landings in klb whole weight

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_klb_yr(iyear)=sum(L_total_klb(iyear));
        L_total_knum_yr(iyear)=sum(L_total_num(iyear))/1000.0;

        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_MRFSS_D_knum(iyear);
        D_total_klb_yr(iyear)+=pred_MRFSS_D_klb(iyear);

        B(iyear)=elem_prod(N(iyear),wgt_mt);
        totN(iyear)=sum(N(iyear));
        totB(iyear)=sum(B(iyear));
  }
  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);
    }
  if(SSB_msy_out>0)
    {
      SdSSB_msy=SSB/SSB_msy_out;
      SdSSB_msy_end=SdSSB_msy(endyr);
    }

   //fill in log recruitment deviations for yrs they are nonzero
   for(iyear=styr_rec_dev; iyear<=endyr; iyear++)
   {
    log_rec_dev_output(iyear)=log_rec_dev(iyear);
   }
   //fill in log Nage deviations for ages they are nonzero (ages2+)
   for(iage=2; iage<=nages; iage++)
   {
    log_Nage_dev_output(iage)=log_Nage_dev(iage);
   }
```

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

  //static per-recruit stuff

  for(iyear=styr; iyear<=endyr; iyear++)
  {
    N_age_spr(1)=1.0;
    for(iage=2; iage<=nages; iage++)
    {
      N_age_spr(iage)=N_age_spr(iage-1)*mfexp(-1.*Z(iyear,iage-1));
    }
    N_age_spr(nages)=N_age_spr(nages)/(1.0-mfexp(-1.*Z(iyear,nages)));
    N_age_spr_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(int 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_RVC_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
      neff_CVT_lenc_allyr_out=missing;
      neff_cL_lenc_allyr_out=missing;
      neff_cO_lenc_allyr_out=missing;
      neff_HB_lenc_allyr_out=missing;
      neff_HB_D_lenc_allyr_out=missing;
      neff_MRFSS_lenc_allyr_out=missing;
      neff_CVT_agec_allyr_out=missing;
      neff_cL_agec_allyr_out=missing;
      neff_HB_agec_allyr_out=missing;
      neff_MRFSS_agec_allyr_out=missing;

      for (iyear=styr_RVC_lenc; iyear<=endyr_RVC_lenc; iyear++)
         {if (nsamp_RVC_lenc(iyear)>=minSS_RVC_lenc)
            { numer=sum( elem_prod(pred_RVC_lenc(iyear),(1.0-pred_RVC_lenc(iyear))) );
              denom=sum( square(obs_RVC_lenc(iyear)-pred_RVC_lenc(iyear)) );
                if (denom>0.0) {neff_RVC_lenc_allyr_out(iyear)=numer/denom;}
                else {neff_RVC_lenc_allyr_out(iyear)=-missing;}
            } else {neff_RVC_lenc_allyr_out(iyear)=-99;}
         }

      for (iyear=styr_CVT_lenc; iyear<=endyr_CVT_lenc; iyear++)
         {if (nsamp_CVT_lenc(iyear)>=minSS_CVT_lenc)
            { numer=sum( elem_prod(pred_CVT_lenc(iyear),(1.0-pred_CVT_lenc(iyear))) );
              denom=sum( square(obs_CVT_lenc(iyear)-pred_CVT_lenc(iyear)) );
                if (denom>0.0) {neff_CVT_lenc_allyr_out(iyear)=numer/denom;}
                else {neff_CVT_lenc_allyr_out(iyear)=-missing;}
            } else {neff_CVT_lenc_allyr_out(iyear)=-99;}
         }

      for (iyear=styr_cL_lenc; iyear<=endyr_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(iyear)=numer/denom;}
                else {neff_cL_lenc_allyr_out(iyear)=-missing;}
            } else {neff_cL_lenc_allyr_out(iyear)=-99;}
         }

      for (iyear=1; iyear<=nyr_cO_lenc; iyear++)
         {if (nsamp_cO_lenc(iyear)>=minSS_cO_lenc)
            { numer=sum( elem_prod(pred_cO_lenc(iyear),(1.0-pred_cO_lenc(iyear))) );
              denom=sum( square(obs_cO_lenc(iyear)-pred_cO_lenc(iyear)) );
                if (denom>0.0) {neff_cO_lenc_allyr_out(yrs_cO_lenc(iyear))=numer/denom;}
                else {neff_cO_lenc_allyr_out(yrs_cO_lenc(iyear))=-missing;}
            } else {neff_cO_lenc_allyr_out(yrs_cO_lenc(iyear))=-99;}
         }

      for (iyear=styr_HB_lenc; iyear<=endyr_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(iyear)=numer/denom;}
                else {neff_HB_lenc_allyr_out(iyear)=-missing;}
            } else {neff_HB_lenc_allyr_out(iyear)=-99;}
         }

      for (iyear=styr_HB_D_lenc; iyear<=endyr_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(iyear)=numer/denom;}
                else {neff_HB_D_lenc_allyr_out(iyear)=-missing;}
            } else {neff_HB_D_lenc_allyr_out(iyear)=-99;}
         }

      for (iyear=styr_MRFSS_lenc; iyear<=endyr_MRFSS_lenc; iyear++)
         {if (nsamp_MRFSS_lenc(iyear)>=minSS_MRFSS_lenc)
            { numer=sum( elem_prod(pred_MRFSS_lenc(iyear),(1.0-pred_MRFSS_lenc(iyear))) );
              denom=sum( square(obs_MRFSS_lenc(iyear)-pred_MRFSS_lenc(iyear)) );
                if (denom>0.0) {neff_MRFSS_lenc_allyr_out(iyear)=numer/denom;}
                else {neff_MRFSS_lenc_allyr_out(iyear)=-missing;}
            } else {neff_MRFSS_lenc_allyr_out(iyear)=-99;}
         }


      for (iyear=styr_CVT_agec; iyear<=endyr_CVT_agec; iyear++)
         {if (nsamp_CVT_agec(iyear)>=minSS_CVT_agec)
            { numer=sum( elem_prod(pred_CVT_agec(iyear),(1.0-pred_CVT_agec(iyear))) );
              denom=sum( square(obs_CVT_agec(iyear)-pred_CVT_agec(iyear)) );
                if (denom>0.0) {neff_CVT_agec_allyr_out(iyear)=numer/denom;}
                else {neff_CVT_agec_allyr_out(iyear)=-missing;}
            } else {neff_CVT_agec_allyr_out(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_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=styr_MRFSS_agec; iyear<=endyr_MRFSS_agec; iyear++)
         {if (nsamp_MRFSS_agec(iyear)>=minSS_MRFSS_agec)
            { numer=sum( elem_prod(pred_MRFSS_agec(iyear),(1.0-pred_MRFSS_agec(iyear))) );
              denom=sum( square(obs_MRFSS_agec(iyear)-pred_MRFSS_agec(iyear)) );
                if (denom>0.0) {neff_MRFSS_agec_allyr_out(iyear)=numer/denom;}
                else {neff_MRFSS_agec_allyr_out(iyear)=-missing;}
            } else {neff_MRFSS_agec_allyr_out(iyear)=-99;}
         }



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


FUNCTION evaluate_objective_function
  fval=0.0;
  fval_unwgt=0.0;

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


  //cout<<fval<<endl;
////---Indices-------------------------------
  f_RVC_cpue=0.0;
  for (iyear=styr_RVC_cpue; iyear<=endyr_RVC_cpue; iyear++)
  {
    f_RVC_cpue+=square(log((pred_RVC_cpue(iyear)+dzero)/
        (obs_RVC_cpue(iyear)+dzero)))/(2.0*log(1.0+square(RVC_cpue_cv(iyear))));
  }
  fval+=w_I_RVC*f_RVC_cpue;
  fval_unwgt+=f_RVC_cpue;

  f_CVT_cpue=0.0;
  for (iyear=styr_CVT_cpue; iyear<=endyr_CVT_cpue; iyear++)
  {
    f_CVT_cpue+=square(log((pred_CVT_cpue(iyear)+dzero)/
        (obs_CVT_cpue(iyear)+dzero)))/(2.0*log(1.0+square(CVT_cpue_cv(iyear))));
  }

  fval+=w_I_CVT*f_CVT_cpue;
  fval_unwgt+=f_CVT_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))));
  }
  fval+=w_I_cL*f_cL_cpue;
  fval_unwgt+=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))));
  }
  fval+=w_I_HB*f_HB_cpue;
  fval_unwgt+=f_HB_cpue;

  f_MRFSS_cpue=0.0;
  for (iyear=styr_MRFSS_cpue; iyear<=endyr_MRFSS_cpue; iyear++)
  {
    f_MRFSS_cpue+=square(log((pred_MRFSS_cpue(iyear)+dzero)/
        (obs_MRFSS_cpue(iyear)+dzero)))/(2.0*log(1.0+square(MRFSS_cpue_cv(iyear))));
  }
  fval+=w_I_MRFSS*f_MRFSS_cpue;
  fval_unwgt+=f_MRFSS_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))));
  }
  fval+=w_L*f_cL_L;
  fval_unwgt+=f_cL_L;

  f_cO_L=0.0; //in 1000 lb ww
  for (iyear=styr_cO_L; iyear<=endyr_cO_L; iyear++)
  {
    f_cO_L+=square(log((pred_cO_L_klb(iyear)+dzero)/
        (obs_cO_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*cO_L_cv(iyear))));
  }
  fval+=w_L*f_cO_L;
  fval_unwgt+=f_cO_L;

  f_HB_L=0.0; //in 1000 fish
  for (iyear=styr_HB_L; 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))));
  }
  fval+=w_L*f_HB_L;
  fval_unwgt+=f_HB_L;

  f_MRFSS_L=0.0; //in 1000 fish
  for (iyear=styr_MRFSS_L; iyear<=endyr_MRFSS_L; iyear++)
  {
    f_MRFSS_L+=square(log((pred_MRFSS_L_knum(iyear)+dzero)/
        (L_mrfss_bias*obs_MRFSS_L(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*MRFSS_L_cv(iyear))));
  }
  fval+=w_L*f_MRFSS_L;
  fval_unwgt+=f_MRFSS_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))));
  }
  fval+=w_D*f_cL_D;
  fval_unwgt+=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))));
  }
  fval+=w_D*f_HB_D;
  fval_unwgt+=f_HB_D;

  f_MRFSS_D=0.0; //in 1000 fish
  for (iyear=styr_MRFSS_D; iyear<=endyr_MRFSS_D; iyear++)
  {
    f_MRFSS_D+=square(log((pred_MRFSS_D_knum(iyear)+dzero)/
        (obs_MRFSS_D(iyear)+dzero)))/(2.0*log(1.0+square(LD_cv_adj*MRFSS_D_cv(iyear))));
  }
  fval+=w_D*f_MRFSS_D;
  fval_unwgt+=f_MRFSS_D;


////---Length comps-------------------------------
  f_RVC_lenc=0.0;
  for (iyear=styr_RVC_lenc; iyear<=endyr_RVC_lenc; iyear++)
  {
    if (nsamp_RVC_lenc(iyear)>=minSS_RVC_lenc)
    {
    f_RVC_lenc-=neff_RVC_lenc(iyear)*
        sum( elem_prod((obs_RVC_lenc(iyear)+dzero),
            log(elem_div((pred_RVC_lenc(iyear)+dzero),
               (obs_RVC_lenc(iyear)+dzero)))));
    }
  }
  fval+=w_lc*f_RVC_lenc;
  fval_unwgt+=f_RVC_lenc;


  f_CVT_lenc=0.0;
  for (iyear=styr_CVT_lenc; iyear<=endyr_CVT_lenc; iyear++)
  {
    if (nsamp_CVT_lenc(iyear)>=minSS_CVT_lenc)
    {
    f_CVT_lenc-=neff_CVT_lenc(iyear)*
        sum( elem_prod((obs_CVT_lenc(iyear)+dzero),
            log(elem_div((pred_CVT_lenc(iyear)+dzero),
               (obs_CVT_lenc(iyear)+dzero)))));
    }
  }
  fval+=w_lc*f_CVT_lenc;
  fval_unwgt+=f_CVT_lenc;


  f_cL_lenc=0.0;
  for (iyear=styr_cL_lenc; iyear<=endyr_cL_lenc; iyear++)
  {
    if (nsamp_cL_lenc(iyear)>=minSS_cL_lenc)
    {
    f_cL_lenc-=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+=w_lc*f_cL_lenc;
  fval_unwgt+=f_cL_lenc;


  f_cO_lenc=0.;
  for (iyear=1; iyear<=nyr_cO_lenc; iyear++)
  {
    if (nsamp_cO_lenc(iyear)>=minSS_cO_lenc)
    {
    f_cO_lenc-=neff_cO_lenc(iyear)*
         sum(elem_prod((obs_cO_lenc(iyear)+dzero),
             log(elem_div((pred_cO_lenc(iyear)+dzero),
                (obs_cO_lenc(iyear)+dzero)))));
    }
  }
  fval+=w_lc*f_cO_lenc;
  fval_unwgt+=f_cO_lenc;


  f_HB_lenc=0.0;
  for (iyear=styr_HB_lenc; iyear<=endyr_HB_lenc; iyear++)
  {
    if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
    {
    f_HB_lenc-=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+=w_lc*f_HB_lenc;
  fval_unwgt+=f_HB_lenc;


  f_HB_D_lenc=0.0;
  for (iyear=styr_HB_D_lenc; iyear<=endyr_HB_D_lenc; iyear++)
  {
    if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
    {
    f_HB_D_lenc-=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+=w_lc*f_HB_D_lenc;
  fval_unwgt+=f_HB_D_lenc;


  f_MRFSS_lenc=0.0;
  for (iyear=styr_MRFSS_lenc; iyear<=endyr_MRFSS_lenc; iyear++)
  {
    if (nsamp_MRFSS_lenc(iyear)>=minSS_MRFSS_lenc)
    {
    f_MRFSS_lenc-=neff_MRFSS_lenc(iyear)*
        sum( elem_prod((obs_MRFSS_lenc(iyear)+dzero),
            log(elem_div((pred_MRFSS_lenc(iyear)+dzero),
               (obs_MRFSS_lenc(iyear)+dzero)))));
    }
  }
  fval+=w_lc*f_MRFSS_lenc;
  fval_unwgt+=f_MRFSS_lenc;


//////---Age comps-------------------------------
  f_CVT_agec=0.0;
  for (iyear=styr_CVT_agec; iyear<=endyr_CVT_agec; iyear++)
  {
    if (nsamp_CVT_agec(iyear)>=minSS_CVT_agec)
    {
    f_CVT_agec-=neff_CVT_agec(iyear)*
        sum(elem_prod((obs_CVT_agec(iyear)+dzero),
            log(elem_div((pred_CVT_agec(iyear)+dzero),
               (obs_CVT_agec(iyear)+dzero)))));
    }
  }
  fval+=w_ac*f_CVT_agec;
  fval_unwgt+=f_CVT_agec;


  f_cL_agec=0.0;
  for (iyear=1; iyear<=nyr_cL_agec; iyear++)
  {
    if (nsamp_cL_agec(iyear)>=minSS_cL_agec)
    {
    f_cL_agec-=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+=w_ac*f_cL_agec;
  fval_unwgt+=f_cL_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-=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+=w_ac*f_HB_agec;
  fval_unwgt+=f_HB_agec;


  f_MRFSS_agec=0.0;
  for (iyear=styr_MRFSS_agec; iyear<=endyr_MRFSS_agec; iyear++)
  {
    if (nsamp_MRFSS_agec(iyear)>=minSS_MRFSS_agec)
    {
    f_MRFSS_agec-=neff_MRFSS_agec(iyear)*
        sum(elem_prod((obs_MRFSS_agec(iyear)+dzero),
            log(elem_div((pred_MRFSS_agec(iyear)+dzero),
                (obs_MRFSS_agec(iyear)+dzero)))));
    }
  }
  fval+=w_ac*f_MRFSS_agec;
  fval_unwgt+=f_MRFSS_agec;


////-----------Constraints and penalties--------------------------------
  f_rec_dev=0.0;
  f_rec_dev=norm2(log_rec_dev);
  f_rec_dev=pow(log_rec_dev(styr_rec_dev),2);
  for(iyear=(styr_rec_dev+1); iyear<=endyr; iyear++)
  {f_rec_dev+=pow(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1),2);}
  fval+=w_rec*f_rec_dev;


  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_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_MRFSS_RW_cpue=0.0;
  for (iyear=styr_MRFSS_cpue; iyear<endyr_MRFSS_cpue; iyear++)
      {f_MRFSS_RW_cpue+=square(q_RW_log_dev_MRFSS(iyear))/(2.0*set_q_RW_MRFSS_var);}
  fval+=f_MRFSS_RW_cpue;

  f_priors=0.0;
  f_priors=norm2(log_Nage_dev);
  f_priors+=square(mfexp(log_len_cv)-set_len_cv)/square(set_len_cv_se);
  f_priors+=square(steep-set_steep)/square(set_steep_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(selpar_L50_cL2-set_selpar_L50_cL2);
  f_priors+=square(selpar_slope_cL2-set_selpar_slope_cL2);
//  f_priors+=square(selpar_L50_cL3-set_selpar_L50_cL3);
  f_priors+=square(selpar_slope_cL3-set_selpar_slope_cL3);

//  f_priors+=square(selpar_L50_cO2-set_selpar_L50_cO2);
  f_priors+=square(selpar_slope_cO2-set_selpar_slope_cO2);
//  f_priors+=square(selpar_L502_cO2-set_selpar_L502_cO2);
  f_priors+=square(selpar_slope2_cO2-set_selpar_slope2_cO2);

//  f_priors+=square(selpar_L50_HB1-set_selpar_L50_HB1);
  f_priors+=square(selpar_slope_HB1-set_selpar_slope_HB1);
//  f_priors+=square(selpar_L50_HB2-set_selpar_L50_HB2);
  f_priors+=square(selpar_slope_HB2-set_selpar_slope_HB2);
//  f_priors+=square(selpar_L50_HB3-set_selpar_L50_HB3);
  f_priors+=square(selpar_slope_HB3-set_selpar_slope_HB3);

//  f_priors+=square(selpar_Age1_HB_D3-set_selpar_Age1_HB_D3);

//  f_priors+=square(selpar_L50_MRFSS3-set_selpar_L50_MRFSS3);
  f_priors+=square(selpar_slope_MRFSS3-set_selpar_slope_MRFSS3);

//  f_priors+=square(selpar_L50_CVT-set_selpar_L50_CVT);
  f_priors+=square(selpar_slope_CVT-set_selpar_slope_CVT);
//  f_priors+=square(selpar_L502_CVT-set_selpar_L502_CVT);
//  f_priors+=square(selpar_slope2_CVT-set_selpar_slope2_CVT);

//  f_priors+=square(selpar_L50_RVC-set_selpar_L50_RVC);
  f_priors+=square(selpar_slope_RVC-set_selpar_slope_RVC);
//  f_priors+=square(selpar_L502_RVC-set_selpar_L502_RVC);
//  f_priors+=square(selpar_slope2_RVC-set_selpar_slope2_RVC);

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


REPORT_SECTION
//  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;

//    cout<< mfexp(log_len_cv)<<endl;
//  report << "TotalLikelihood " << fval << endl;
 #include "rg_make_Robject1.cxx"   // write the S-compatible report
```

## Appendix B: AD Model Builder data input file for red grouper

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##  Data Input File
##  SEDAR19 Assessment: Red Grouper
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#starting and ending year of model
1976
2008
#Starting year to estimate recruitment deviation from S-R curve
1976
#3 phases of constraints on recruitment deviations: allows possible heavier constraint in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1977
```

```
2006
#3 periods of size regs: yr1-83 no restrictions, 1984-91 12-inch TL, 1992-09 20-in TL
#ending years of regulation period
1983
1991
##Size limits of periods 2, 3 (in mm: 1mm=0.0394in)
304.5685 #period 2
507.6142 #period 3
406.0914 #release size applied to discards in period 2: based on FL regs

#Number of ages (20 classes is 1,...,20+)
16
##vector of agebins, last is a plus group
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
##number length bins used to match length comps and number used to compute plus group
35
11
#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
160 190 220 250 280 310 340 370 400 430 460 490 520 550 580 610 640 670 700 730 760 790 820 850 880 910 940 970 1000    1030    1060    1090 1120 1150  1180
1210 1240   1270    1300    1330    1360    1390    1420    1450    1480    1510
#discard mortality constant
0.2  #comm HAL
0.2  #headboat
0.2  #MRFSS
#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
#VonBert params (Linf, K, t0), units in mm TL
848.21
0.213
-0.67
#Standard errors of vonBert param (Linf, K, t0), applied if params are estimated
4.412
0.003
0.037
#cv of length at age
0.09
#standard error of cv of length at age, applied if cv is estimated
0.0008
#length-weight (TL-whole wgt) coefficients a and b, W=aL^b, (W in g, TL in mm)
8.418E-06
3.1
##time-invariant vector of % maturity-at-age for females (ages 1-20)
0.0 0.354   0.539   0.714   0.842   0.919   0.960   0.981   0.991   0.996   0.998   1.0 1.0 1.0 1.0 1.0
##time-invariant vector of % maturity-at-age for males (ages 1-20)
0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
#time-variant vector of proportion female (ages 1-20)
1.0 0.963   0.930   0.878   0.803   0.705   0.590   0.466   0.346   0.239   0.153   0.091   0.050   0.025   0.0 0.0
#time of year (as fraction) for spawning: mid-April=115d/365d
0.315
#age-dependent natural mortality at age
0.30    0.23    0.19    0.17    0.16    0.15    0.14    0.13    0.13    0.13    0.13    0.12    0.12    0.12    0.12    0.12
#age-independent natural mortality (used only to compute MSST=(1-M)SSBmsy)
0.14
##Spawner-recruit parameters
#steepness (fixed or initial guess) (0.72 from meta-analysis)
0.72
#standard error of steepness (from meta-analysis)
0.17
#log_R0 - log virgin recruitment
12.9
# R autocorrelation
0.0
###################################################################################################
####FL Keys Visual Survey (RVC)####################################################################
##Starting and ending years of time series, respectively
1994
2008
##Observed CPUE (numbers) and CV vectors, respectively
0.38    0.29    0.35    0.42    0.74    1.70    1.55    1.68    1.37    1.53    1.16    1.30    0.63    0.67    1.24
0.65        0.28        0.42        0.42        0.53        0.3         0.13        0.11        0.12        0.12        0.21        0.12        0.17        0.15        0.11
#Starting and ending year of length composition data
1994
2008
#sample sizes of length comps by year   (first row observed N, second row effective N: effective may be set to observed)
126.0   291.0   151.0   408.0   461.0   440.0   527.0   742.0   628.0   448.0   246.0   498.0   608.0   619.0   735.0
126.0   291.0   151.0   408.0   461.0   440.0   527.0   742.0   628.0   448.0   246.0   498.0   608.0   619.0   735.0
#length composition by year (year,lengthbin 3cm)
0.1674  0.1573  0.1752  0.0730  0.0100  0.0398  0.0512  0.0295  0.0574  0.0423  0.0590  0.0571  0.0587  0.0209  0.0011  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0191  0.0313  0.0881  0.1243  0.0944  0.1233  0.1448  0.1872  0.1329  0.0470  0.0064  0.0012  0.0001  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.1021  0.0706  0.0537  0.0639  0.1207  0.0944  0.1372  0.1743  0.0787  0.0472  0.0374  0.0183  0.0017  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0101  0.0557  0.0822  0.1037  0.1036  0.1514  0.1135  0.1086  0.0930  0.0463  0.0574  0.0310  0.0041  0.0006  0.0000  0.0010  0.0112  0.0203  0.0058  0.0003  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0184  0.0756  0.1772  0.1527  0.1161  0.1152  0.1042  0.0933  0.0664  0.0412  0.0161  0.0106  0.0096  0.0032  0.0002  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.1742  0.1436  0.1035  0.1029  0.0746  0.0840  0.0801  0.0465  0.0594  0.0526  0.0142  0.0175  0.0154  0.0085  0.0070  0.0017  0.0022  0.0075  0.0041  0.0004  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0341  0.0670  0.0687  0.0880  0.1070  0.1646  0.1437  0.0966  0.0747  0.0627  0.0343  0.0207  0.0122  0.0052  0.0036  0.0022  0.0049  0.0026  0.0002  0.0000  0.0000   0.0000 0.0000  0.0010  0.0036  0.0021  0.0
0.0173  0.0232  0.0316  0.0395  0.0521  0.0882  0.1427  0.1492  0.0985  0.0781  0.0767  0.0563  0.0371  0.0168  0.0099  0.0130  0.0176  0.0116  0.0083  0.0100  0.0036   0.0016 0.0080  0.0071  0.0017  0.0004  0.0
0.0113  0.0185  0.0241  0.0433  0.0625  0.0950  0.0946  0.0780  0.0774  0.0814  0.0727  0.0524  0.0509  0.0409  0.0345  0.0369  0.0282  0.0177  0.0137  0.0138  0.0217   0.0247 0.0055  0.0002  0.0000  0.0000  0.0
0.0195  0.0266  0.0451  0.0699  0.0719  0.0787  0.0698  0.0644  0.0744  0.0679  0.0576  0.0501  0.0475  0.0279  0.0234  0.0245  0.0198  0.0103  0.0121  0.0247  0.0272   0.0260 0.0140  0.0083  0.0035  0.0071  0.0
0.0187  0.0216  0.0250  0.0308  0.0872  0.0885  0.0626  0.0861  0.1201  0.1072  0.0664  0.0341  0.0179  0.0117  0.0106  0.0112  0.0216  0.0510  0.0337  0.0157  0.0137   0.0147 0.0130  0.0089  0.0018  0.0067  0.0
0.0040  0.0231  0.0296  0.0284  0.0431  0.0890  0.0920  0.0954  0.1104  0.1074  0.0924  0.0715  0.0670  0.0385  0.0263  0.0322  0.0238  0.0120  0.0090  0.0030  0.0002   0.0001 0.0008  0.0006  0.0001  0.0000  0.0
0.0370  0.0543  0.0494  0.0341  0.0282  0.0371  0.0760  0.1117  0.1242  0.1277  0.0892  0.0494  0.0285  0.0227  0.0296  0.0357  0.0192  0.0063  0.0065  0.0105  0.0104   0.0098 0.0024  0.0001  0.0000  0.0000  0.0
0.0080  0.0106  0.0227  0.0346  0.0535  0.0846  0.0784  0.0798  0.1108  0.0914  0.0465  0.0419  0.0457  0.0334  0.0330  0.0555  0.0507  0.0559  0.0303  0.0061  0.0091   0.0121 0.0028  0.0001  0.0001  0.0006  0.0
0.0065  0.0217  0.0452  0.0676  0.0759  0.1095  0.1273  0.1024  0.0676  0.0653  0.0599  0.0506  0.0369  0.0217  0.0271  0.0406  0.0294  0.0134  0.0088  0.0039  0.0056   0.0091 0.0038  0.0003  0.0000  0.0000  0.0
###################################################################################################
#
```

```
#################MARMAP Chevron trap index##########################################################
##Starting and ending years of time series, respectively
1990
2008
##Observed CPUE (numbers) and CV vectors, respectively
0.05    0.12    0.45    0.86    1.1 0.52    2.61    0.32    0.33    1.7 1.13    1.7 1.16    1.02    1.03    1.15    1.7 1.51    0.56
1.13    0.94    0.75    0.65    0.62    0.76    0.61    0.88    0.65    0.53    0.49    0.53    0.52    0.52    0.53    0.48    0.52    0.49    0.66
##Starting and ending year of length composition data
1997
2007
##sample sizes of length comps by year   (first row observed N, second row effective N: effective may be set to observed)
23.0    28.0    21.0    25.0    20.0    21.0    19.0    22.0    25.0    18.0    21.0
23.0    28.0    21.0    25.0    20.0    21.0    19.0    22.0    25.0    18.0    21.0
##length composition samples (year,lengthbin 1cm)
0.0000  0.0000  0.0000  0.0000  0.0000  0.0250  0.1000  0.0500  0.1250  0.2000  0.0750  0.1750  0.0750  0.0250  0.0000  0.0250  0.0000  0.0250  0.0750  0.0000  0.0000   0.0250 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0256  0.0256  0.0641  0.2435  0.3206  0.0770  0.0769  0.0769  0.0000  0.0256  0.0000  0.0384  0.0128  0.0128   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0833  0.0208  0.0000  0.0625  0.2292  0.2083  0.1041  0.0834  0.0625  0.1042  0.0208  0.0000  0.0000   0.0208 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0263  0.0000  0.0000  0.0526  0.0263  0.0263  0.0789  0.2894  0.2368  0.1579  0.1052  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0526  0.1842  0.1315  0.0789  0.0526  0.0263  0.0263  0.0263  0.0526  0.0263  0.0526  0.1052  0.1053  0.0526  0.0263   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0270  0.0811  0.0270  0.0270  0.0540  0.1081  0.2163  0.2433  0.0811  0.0000  0.0270  0.0000  0.0000  0.0541  0.0270  0.0270   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.1081  0.0000  0.0541  0.0810  0.1621  0.0541  0.0811  0.0811  0.0811  0.1081  0.1082  0.0540  0.0000  0.0000  0.0000  0.0000   0.0270 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.1250  0.1750  0.0250  0.0250  0.0750  0.0500  0.0750  0.1500  0.1500  0.0250  0.0500  0.0250  0.0000  0.0250  0.0000  0.0000  0.0000   0.0250 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0690  0.0000  0.1724  0.1035  0.0345  0.0345  0.0345  0.0345  0.0690  0.1035  0.1724  0.1379  0.0000  0.0000  0.0000   0.0345 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0227  0.0455  0.1819  0.2955  0.0455  0.0455  0.0455  0.0454  0.0455  0.0227  0.1592  0.0227  0.0000  0.0000   0.0000 0.0227  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0233  0.0000  0.0000  0.1163  0.0930  0.3023  0.1861  0.0931  0.0233  0.0233  0.0233  0.0000  0.0466  0.0233   0.0233 0.0233  0.0000  0.0000  0.0000  0.0
##Starting and ending years of age composition data
1997
2007
#sample sizes of age comps by year   (first row observed N, second row effective N: effective may be set to observed)
23.0    28.0    21.0    25.0    20.0    21.0    19.0    22.0    25.0    18.0    21.0
23.0    28.0    21.0    25.0    20.0    21.0    19.0    22.0    25.0    18.0    21.0
#age composition samples (year,age)
0.0000  0.1111  0.3611  0.3611  0.0000  0.0278  0.1111  0.0000  0.0000  0.0000  0.0278  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0694  0.6111  0.2361  0.0139  0.0556  0.0139  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0851  0.3830  0.4255  0.0851  0.0000  0.0213  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0303  0.0606  0.3636  0.3030  0.1818  0.0606  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.4595  0.0811  0.0811  0.1351  0.2432  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.1111  0.0833  0.6111  0.0833  0.0000  0.0556  0.0278  0.0278  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.1111  0.2778  0.1667  0.4167  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0278  0.0000  0.0000
0.0000  0.2308  0.2308  0.4103  0.0513  0.0513  0.0000  0.0000  0.0256  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.2857  0.2143  0.3571  0.0357  0.0714  0.0000  0.0000  0.0000  0.0357  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.6136  0.1364  0.2045  0.0227  0.0000  0.0000  0.0000  0.0000  0.0227  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0233  0.0233  0.7674  0.1163  0.0233  0.0000  0.0000  0.0000  0.0000  0.0000  0.0465  0.0000  0.0000  0.0000

####################################################################################################
#
#####################Commercial Hook and Line fishery landings##########################
##Commercial Hook and Line CPUE Index from Logbook
##Starting and ending years of CPUE index
1993
2008
##Observed CPUE and assumed CVs
0.39    0.31    0.50    0.58    0.65    0.99    1.45    1.05    0.84    0.90    1.03    0.96    0.86    1.22    1.94    2.35
0.27    0.26    0.21    0.17    0.15    0.12    0.10    0.13    0.15    0.15    0.14    0.14    0.15    0.13    0.11    0.10
##Commercial Hook and Line fishery landings
#Starting and ending years of landings time series, respectively
1976
2008
##Observed landings (1000 lb whole weight) and assumed CVs
263.678 209.245 257.966 234.447 184.857 210.664 205.599 203.609 236.620 201.470 249.957 189.755 244.353 230.244 172.989 139.206 128.888 168.202 165.351 230.109 279.453   310.997    431.654 404.755 342.501 327.783
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01     0.01   0.01    0.01    0.01    0.01    0.0
##Starting and ending years of discards time series, respectively
1992
2008
##Observed discards (1000 fish) and assumed CVs
8.915   8.575   14.397  10.489  11.582  14.709  10.461  12.956  10.869  8.423   21.608  11.354  10.850  9.992   4.933   8.571   1.993
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01
##Starting and ending years of commercial hook and line length composition sample data
1986 #1986 starts TIP in FL
2008
#sample size of commercial length comp data by year    (first row observed N, second row effective N: effective may be set to observed)
33.0    46.0    67.0    62.0    73.0    77.0    42.0    71.0    76.0    94.0    79.0    64.0    105.0   176.0   214.0   149.0   116.0   135.0   181.0   259.0   314.0    432.0  408.0
33.0    46.0    67.0    62.0    73.0    77.0    42.0    71.0    76.0    94.0    79.0    64.0    105.0   176.0   214.0   149.0   116.0   135.0   181.0   259.0   314.0    432.0  408.0
#commercial length composition samples (year,lengthbin 3 cm)
0.0000  0.0000  0.0000  0.0000  0.0010  0.0104  0.1291  0.1251  0.0983  0.0950  0.1016  0.0817  0.0586  0.0562  0.0706  0.0577  0.0358  0.0343  0.0157  0.0113  0.0088   0.0057 0.0001  0.0031  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0033  0.0318  0.1076  0.1345  0.1300  0.0982  0.1014  0.0927  0.0777  0.0731  0.0402  0.0378  0.0293  0.0202  0.0124  0.0036  0.0025   0.0013 0.0022  0.0001  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0166  0.1486  0.2011  0.1464  0.1095  0.0937  0.0692  0.0533  0.0383  0.0359  0.0295  0.0208  0.0136  0.0098  0.0059  0.0026  0.0028   0.0008 0.0009  0.0003  0.0002  0.0001  0.0
0.0000  0.0000  0.0000  0.0000  0.0013  0.0949  0.1481  0.1742  0.1458  0.0972  0.0826  0.0497  0.0497  0.0614  0.0387  0.0249  0.0115  0.0031  0.0044  0.0016  0.0051   0.0031 0.0020  0.0007  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0073  0.0151  0.0175  0.0172  0.0477  0.0904  0.0846  0.0915  0.0981  0.1258  0.1216  0.0751  0.0448  0.0227  0.0372   0.0274 0.0387  0.0242  0.0129  0.0000  0.0
0.0000  0.0000  0.0000  0.0001  0.0000  0.0105  0.0000  0.0234  0.0235  0.0254  0.0240  0.0505  0.0858  0.0838  0.1070  0.1127  0.1157  0.1414  0.0964  0.0461  0.0321   0.0098 0.0118  0.0001  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0039  0.0654  0.1285  0.1117  0.1057  0.1591  0.1344  0.0859  0.0952  0.0427  0.0195   0.0128 0.0195  0.0117  0.0039  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0028  0.0069  0.0069  0.0174  0.0144  0.0316  0.0353  0.0517  0.0860  0.1336  0.1821  0.1273  0.1253  0.0735  0.0407   0.0406 0.0211  0.0028  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0040  0.0119  0.0093  0.0119  0.1044  0.1110  0.1063  0.0811  0.0813  0.1291  0.1176  0.1027  0.0500  0.0336   0.0296 0.0095  0.0066  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0010  0.0042  0.0038  0.0209  0.0325  0.0666  0.1764  0.2504  0.1637  0.0921  0.0731  0.0544  0.0315   0.0216 0.0015  0.0053  0.0010  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0050  0.0050  0.0050  0.0112  0.0039  0.0421  0.0898  0.0982  0.1549  0.1878  0.1823  0.1001  0.0528  0.0269   0.0177 0.0121  0.0051  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0025  0.0714  0.1309  0.0788  0.0533  0.0967  0.1093  0.2121  0.1371  0.0528  0.0260   0.0283 0.0008  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0029  0.0081  0.0119  0.0607  0.1637  0.1219  0.1346  0.1498  0.0842  0.0919  0.0670  0.0539  0.0303   0.0057 0.0109  0.0026  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0007  0.0015  0.0128  0.0431  0.0762  0.1647  0.2431  0.1939  0.1125  0.0649  0.0472  0.0245   0.0107 0.0041  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0025  0.0000  0.0000  0.0006  0.0008  0.0038  0.0157  0.0494  0.0793  0.1934  0.2358  0.1952  0.1237  0.0559  0.0259   0.0116 0.0033  0.0031  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0020  0.0020  0.0000  0.0090  0.0374  0.0070  0.0219  0.0374  0.1074  0.2054  0.2871  0.1925  0.0579   0.0252 0.0051  0.0024  0.0004  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0027  0.0330  0.0419  0.0392  0.0265  0.0142  0.0220  0.0941  0.1871  0.2547  0.1643   0.0796 0.0404  0.0004  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0025  0.0027  0.0107  0.0576  0.0566  0.0592  0.0836  0.0677  0.0511  0.0382  0.0917  0.1590  0.1876   0.1045 0.0199  0.0073  0.0002  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0009  0.0018  0.0031  0.0074  0.0626  0.1021  0.0983  0.0639  0.1086  0.1178  0.1106  0.0617  0.0536  0.0786   0.0903 0.0281  0.0065  0.0040  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0018  0.0343  0.0723  0.1028  0.1103  0.1088  0.0965  0.1104  0.0830  0.0613  0.0767   0.0772 0.0507  0.0136  0.0002  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0005  0.0011  0.0000  0.0048  0.0856  0.1210  0.0908  0.0793  0.0982  0.1262  0.1179  0.0849  0.0665  0.0676   0.0389 0.0139  0.0027  0.0002  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0016  0.0020  0.0012  0.0016  0.0185  0.0682  0.1186  0.1433  0.1369  0.1168  0.1039  0.1111  0.0662  0.0537   0.0372 0.0102  0.0052  0.0035  0.0004  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0010  0.0005  0.0000  0.0005  0.0019  0.0157  0.0473  0.0900  0.1956  0.2145  0.1322  0.0909  0.0657  0.0753   0.0459 0.0144  0.0076  0.0011  0.0000  0.0

##Number and vector of years of age compositions for hook and line fishery
5
2004    2005    2006    2007 2008
##sample sizes of age comps by year  (first row observed N, second row effective N: effective may be set to observed)
65.0    147.0   219.0   387.0   395.0
65.0    147.0   219.0   387.0   395.0
```

```
#age composition samples (year,age)
0.002   0.079   0.265   0.212   0.154   0.029   0.063   0.099   0.070   0.010   0.011   0.003   0.000   0.001   0.000   0.000
0.002   0.038   0.244   0.241   0.194   0.075   0.013   0.028   0.066   0.058   0.020   0.006   0.013   0.002   0.000   0.000
0.000   0.020   0.341   0.184   0.276   0.033   0.072   0.006   0.008   0.011   0.027   0.009   0.007   0.000   0.004   0.001
0.000   0.002   0.143   0.437   0.160   0.115   0.042   0.029   0.004   0.016   0.020   0.015   0.008   0.001   0.003   0.003
0.000   0.000   0.020   0.292   0.434   0.077   0.046   0.037   0.018   0.007   0.013   0.027   0.016   0.008   0.001   0.003


#################################Commercial Other (pots + dive + other) fishery landings ###########
##Starting and ending years of landings time series, respectively
1976
2008
#Observed landings (1000 lb whole weight)  and CV's
171.480 135.148 152.356 135.079 103.576 125.994 113.021 118.816 141.385 100.637 130.830 118.235 111.014 113.742 102.009 74.863  39.960  16.477  10.094  9.413   19.121   18.837 35.487  17.033  12.356  43.889  31.
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01     0.01   0.01    0.01    0.01    0.01    0.0
#Number and vector of years of length compositions (comm combined)
3
1986    1987    1989
##sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
13  6  8
13  6  8
#commercial combined gear (data from pots) length comp samples (year,age) (3cm length bins)
0.0000  0.0000  0.0000  0.0000  0.0064  0.0828  0.1623  0.1319  0.1222  0.0828  0.0814  0.0742  0.0485  0.0532  0.0371  0.0258  0.0266  0.0185  0.0193  0.0096  0.0088   0.0056 0.0024  0.0008  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0093  0.1618  0.2216  0.1776  0.1179  0.0677  0.0765  0.0577  0.0302  0.0236  0.0223  0.0196  0.0039  0.0052  0.0026  0.0013  0.0013   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0028  0.1399  0.1848  0.1287  0.1231  0.1201  0.0875  0.0395  0.0368  0.0364  0.0336  0.0250  0.0252  0.0084  0.0083  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0


#########################################Headboat landings#########################################
##Starting and ending years for CPUE index
1978
2008
##Observed CPUE values (numbers) and CVs,
1.78    2.35    0.86    1.04    0.69    1.23    0.76    0.63    0.59    0.83    0.53    0.72    0.61    0.36    0.4     0.5     0.61    0.66    0.74    1.11    1.89     1.58   0.92    1.01    0.76    0.75    1.5
0.2     0.18    0.2     0.21    0.22    0.2     0.2     0.21    0.21    0.21    0.22    0.22    0.23    0.23    0.23    0.22    0.22    0.22    0.21    0.21    0.2      0.21   0.22    0.21    0.22    0.22    0.2
##Starting and ending years for landings time series
1976
2008
##Headboat landings vector (1000 fish) and CV's
4.60    5.61    4.77    9.38    8.14    7.96    6.36    9.89    8.56    8.78    5.81    7.04    5.10    3.62    7.33    2.73    3.98    4.79    5.47    5.25    5.65     8.06   10.90   7.26    5.33    4.94    4.6
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01     0.01   0.01    0.01    0.01    0.01    0.0
##Starting and ending years of discards time series, respectively
2005
2007
##Observed discards (1000s) and assumed CVs
88.18   22.44   20.30
#95.58 43.68 42.99
0.01 0.01 0.01
##Starting and ending year of headboat length composition data
1976
2008
##sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
40.0    73.0    56.0    89.0    115.0   148.0   162.0   224.0   284.0   247.0   203.0   173.0   139.0   153.0   104.0   54.0    59.0    80.0    104.0   112.0   157.0    186.0  272.0   192.0   132.0   119.0   142
40.0    73.0    56.0    89.0    115.0   148.0   162.0   224.0   284.0   247.0   203.0   173.0   139.0   153.0   104.0   54.0    59.0    80.0    104.0   112.0   157.0    186.0  272.0   192.0   132.0   119.0   142
##HB length comp samples (year,lengthbin)
0.0000  0.0000  0.0000  0.0462  0.0000  0.0462  0.0000  0.0154  0.0000  0.0308  0.0154  0.0769  0.1077  0.2000  0.2923  0.0615  0.0000  0.0000  0.0154  0.0000  0.0000   0.0000 0.0000  0.0308  0.0308  0.0154  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0167  0.0083  0.0000  0.0167  0.0000  0.0417  0.0250  0.0583  0.0333  0.1167  0.2333  0.2250  0.0750  0.0167  0.0250   0.0250 0.0083  0.0333  0.0417  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0102  0.0306  0.1020  0.1020  0.0918  0.0408  0.0306  0.0102  0.0306  0.0306  0.0510  0.1122  0.2143  0.1122  0.0000   0.0102 0.0102  0.0102  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0063  0.1076  0.1519  0.0823  0.0696  0.0696  0.0823  0.0570  0.0633  0.0633  0.0127  0.0253  0.0380  0.0443  0.0316  0.0506  0.0316   0.0063 0.0063  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0412  0.1082  0.1134  0.0773  0.0979  0.0722  0.0773  0.0412  0.0567  0.0619  0.0567  0.0464  0.0309  0.0464  0.0155  0.0361   0.0052 0.0155  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0253  0.0380  0.0675  0.0844  0.0970  0.0759  0.0591  0.0675  0.0591  0.0844  0.0886  0.0675  0.0380  0.0422  0.0380  0.0127  0.0084   0.0295 0.0127  0.0000  0.0000  0.0042  0.0
0.0000  0.0000  0.0000  0.0000  0.0072  0.0397  0.1047  0.1191  0.0903  0.0903  0.0794  0.0325  0.0686  0.0722  0.0578  0.0758  0.0542  0.0181  0.0289  0.0144  0.0289   0.0036 0.0108  0.0000  0.0000  0.0036  0.0
0.0000  0.0000  0.0026  0.0000  0.0131  0.0785  0.1728  0.1440  0.0942  0.0550  0.0419  0.0576  0.0288  0.0602  0.0340  0.0419  0.0419  0.0288  0.0183  0.0209  0.0288   0.0131 0.0157  0.0079  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0105  0.0314  0.0820  0.1920  0.1274  0.0942  0.0681  0.0576  0.0733  0.0436  0.0227  0.0506  0.0384  0.0314  0.0297  0.0140  0.0052   0.0140 0.0070  0.0017  0.0017  0.0000  0.0
0.0000  0.0000  0.0000  0.0017  0.0052  0.0260  0.0589  0.0953  0.1196  0.1075  0.0832  0.1057  0.0728  0.0572  0.0780  0.0468  0.0399  0.0243  0.0260  0.0139  0.0121   0.0139 0.0087  0.0000  0.0035  0.0000  0.0
0.0000  0.0000  0.0000  0.0026  0.0000  0.0078  0.0495  0.1094  0.1276  0.0937  0.1224  0.1198  0.1146  0.0599  0.0286  0.0443  0.0443  0.0182  0.0208  0.0078  0.0078   0.0052 0.0026  0.0104  0.0026  0.0000  0.0
0.0000  0.0000  0.0105  0.0035  0.0000  0.0279  0.0453  0.0557  0.0697  0.0801  0.1185  0.1220  0.1359  0.1289  0.0383  0.0592  0.0383  0.0105  0.0383  0.0000  0.0105   0.0035 0.0000  0.0000  0.0000  0.0035  0.0
0.0000  0.0000  0.0000  0.0048  0.0143  0.0333  0.0667  0.0571  0.1286  0.1333  0.1476  0.1143  0.0952  0.0524  0.0333  0.0286  0.0190  0.0238  0.0190  0.0048  0.0048   0.0000 0.0000  0.0048  0.0143  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0041  0.0000  0.0083  0.0207  0.0456  0.0705  0.1577  0.1369  0.1618  0.1245  0.0207  0.1120  0.0415  0.0456  0.0332  0.0083  0.0041   0.0041 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0065  0.0065  0.0065  0.0452  0.0774  0.1161  0.0516  0.1613  0.1484  0.0452  0.1677  0.0968  0.0323  0.0258  0.0000  0.0000   0.0065 0.0000  0.0065  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0154  0.0769  0.0615  0.0923  0.0769  0.2000  0.0923  0.1077  0.0923  0.0923  0.0154  0.0000  0.0462  0.0000   0.0000 0.0154  0.0000  0.0154  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0120  0.0000  0.0241  0.2048  0.2289  0.1566  0.1084  0.0964  0.0723  0.0482  0.0482  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0090  0.0000  0.0090  0.0000  0.0000  0.0180  0.0000  0.0180  0.0991  0.2432  0.1532  0.1171  0.1261  0.1081  0.0541  0.0180  0.0000  0.0090   0.0000 0.0000  0.0180  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0069  0.0069  0.0000  0.0000  0.1250  0.2083  0.2083  0.1042  0.1111  0.0972  0.0625  0.0208  0.0278  0.0139   0.0000 0.0000  0.0069  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0054  0.0054  0.0000  0.0000  0.0161  0.0645  0.1882  0.1774  0.1720  0.1559  0.0968  0.0538  0.0323  0.0215  0.0108   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0076  0.0722  0.2091  0.1407  0.1141  0.0913  0.1027  0.1217  0.1065  0.0228  0.0076   0.0000 0.0000  0.0000  0.0000  0.0038  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0074  0.0050  0.0050  0.0050  0.0099  0.0470  0.1955  0.1040  0.1015  0.1436  0.0941  0.1139  0.1040  0.0421  0.0124   0.0050 0.0000  0.0000  0.0025  0.0000  0.0
0.0000  0.0000  0.0021  0.0000  0.0000  0.0000  0.0000  0.0043  0.0000  0.0000  0.0107  0.1180  0.2618  0.1781  0.1567  0.1094  0.0494  0.0322  0.0258  0.0236  0.0193   0.0043 0.0043  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0065  0.0000  0.0000  0.0032  0.0065  0.0613  0.1742  0.1806  0.0935  0.2032  0.1387  0.0516  0.0290  0.0065  0.0194   0.0194 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0049  0.0000  0.0000  0.0000  0.0000  0.0000  0.0936  0.1823  0.1281  0.0936  0.1429  0.0985  0.1330  0.0936  0.0099  0.0000   0.0099 0.0049  0.0049  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0063  0.0813  0.1938  0.1688  0.1375  0.1063  0.0750  0.0438  0.1000  0.0500  0.0063   0.0250 0.0000  0.0063  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0053  0.0000  0.0000  0.0000  0.0053  0.0794  0.3228  0.2593  0.1429  0.0635  0.0000  0.0317  0.0423  0.0212  0.0000   0.0106 0.0000  0.0000  0.0000  0.0053  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0083  0.1167  0.2167  0.1500  0.1667  0.1583  0.1167  0.0083  0.0250  0.0167  0.0000   0.0083 0.0083  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0066  0.0066  0.0000  0.0000  0.0066  0.0066  0.1316  0.3092  0.2303  0.1053  0.0658  0.0395  0.0329  0.0329  0.0066  0.0000   0.0000 0.0197  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0065  0.1169  0.2078  0.1753  0.1494  0.1039  0.0974  0.0844  0.0065  0.0260  0.0065   0.0000 0.0000  0.0195  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0073  0.1022  0.2482  0.2117  0.1898  0.0803  0.0657  0.0219  0.0365  0.0219  0.0000   0.0000 0.0146  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0159  0.0556  0.1429  0.2381  0.1825  0.0952  0.1190  0.0635  0.0159  0.0079  0.0317   0.0000 0.0079  0.0159  0.0079  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0156  0.0938  0.0938  0.1719  0.1563  0.2031  0.1250  0.0313  0.0469  0.0469   0.0000 0.0156  0.0000  0.0000  0.0000  0.0
##Number and vector of years of age compositions (headboat)
11
1979    1980    1981    1982    1983    1984    2004    2005    2006    2007    2008
##sample sizes of age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
30.0    76.0    113.0   51.0    35.0    41.0    30.0    68.0    52.0    42.0    25.0
30.0    76.0    113.0   51.0    35.0    41.0    30.0    68.0    52.0    42.0    25.0
##headboat age comp samples (year,age)
0.0000  0.3121  0.1210  0.1720  0.2484  0.0573  0.0637  0.0127  0.0064  0.0000  0.0000  0.0064  0.0000  0.0000  0.0000  0.0000
0.0000  0.1423  0.2179  0.1902  0.1625  0.1675  0.0642  0.0327  0.0126  0.0050  0.0050  0.0000  0.0000  0.0000  0.0000  0.0000
0.0443  0.0654  0.2310  0.2300  0.1116  0.1261  0.1376  0.0298  0.0144  0.0077  0.0000  0.0019  0.0000  0.0000  0.0000  0.0000
0.0140  0.0525  0.1979  0.3065  0.2487  0.0788  0.0350  0.0455  0.0000  0.0140  0.0018  0.0053  0.0000  0.0000  0.0000  0.0000
0.0000  0.6781  0.1400  0.0467  0.0369  0.0565  0.0000  0.0074  0.0025  0.0000  0.0049  0.0197  0.0000  0.0074  0.0000  0.0000
0.0382  0.0603  0.6647  0.1368  0.0221  0.0309  0.0088  0.0015  0.0059  0.0059  0.0044  0.0059  0.0088  0.0044  0.0000  0.0015
0.0000  0.0000  0.4722  0.3858  0.0926  0.0494  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0162  0.2282  0.2492  0.4126  0.0356  0.0437  0.0000  0.0049  0.0049  0.0000  0.0049  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0618  0.5792  0.2355  0.0753  0.0193  0.0000  0.0000  0.0000  0.0270  0.0019  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0109  0.2409  0.5839  0.0876  0.0547  0.0000  0.0000  0.0182  0.0000  0.0000  0.0000  0.0036  0.0000  0.0000
```

```
0.0000  0.0078  0.0000  0.1240  0.1473  0.6047  0.0310  0.0775  0.0078  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

##Starting and ending year of headboat discard length composition data
2005
2007
##sample sizes of discard length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
93.0    59.0    36.0
93.0    59.0    36.0
##HB discard length comp samples (year,lengthbin)
0.0000  0.0000  0.0107  0.0178  0.0178  0.0643  0.1106  0.1786  0.1857  0.1786  0.1072  0.1107  0.0143  0.0000  0.0036  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0065  0.0260  0.0520  0.1623  0.1363  0.1753  0.1299  0.1689  0.1235  0.0195  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0158  0.0079  0.0794  0.1031  0.1826  0.2064  0.1190  0.0952  0.1270  0.0397  0.0159  0.0000  0.0000  0.0079  0.0000  0.0000  0.0000  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
#####################################MRFSS landings  #################################
#Recreational MRFSS CPUE Index
#Starting and ending years of CPUE index
1991
2008
#Observed CPUE and assumed CVs
0.30    0.31    0.87    1.09    0.31    1.34    0.94    1.26    0.68    0.77    0.91    1.97    1.33    1.47    1.56    0.81    0.51    1.58
0.51    0.36    0.84    0.37    0.59    0.35    0.61    0.50    0.32    0.33    0.26    0.25    0.26    0.27    0.22    0.27    0.39    0.28
#Recreational landings
#Starting and ending years for landings time series
1981
2008
##MRFSS landings vector (1000 fish) and CVs
79.93   138.64  237.35  206.42  76.69   91.20   80.46   37.99   74.68   12.83   5.95    22.65   50.32   34.43   37.21   46.47   40.68   35.31   19.40   17.63   18.37    39.25  47.95   40.40   35.42   55.80   77.
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01     0.01   0.01    0.01    0.01    0.01    0.0
##Starting and ending years of discards time series, respectively
1981
2008
##Observed discards (1000s) and assumed CVs
15.33   17.47   154.68  175.84  7.19    34.29   114.71  54.63   11.93   21.89   163.80  152.33  79.55   146.42  150.45  344.66  352.94  113.65  110.38  226.80  189.69   122.95 159.81  219.12  230.41  194.77  58.
0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01    0.01     0.01   0.01    0.01    0.01    0.01    0.0
##Starting and ending year of mrfss length composition data
1993
2008
#sample sizes of length comp data by year  (first row observed N, second row effective N: effective may be set to observed)
27.0    27.0    31.0    32.0    29.0    44.0    38.0    24.0    48.0    60.0    53.0    36.0    37.0    40.0    43.0    55.0
27.0    27.0    31.0    32.0    29.0    44.0    38.0    24.0    48.0    60.0    53.0    36.0    37.0    40.0    43.0    55.0
#MRFSS length comp samples (year,lengthbin)
0.0000  0.0000  0.0000  0.0000  0.0289  0.0000  0.0000  0.0000  0.0000  0.0552  0.0252  0.0240  0.1096  0.1294  0.4467  0.0461  0.0359  0.0078  0.0000  0.0000  0.0912   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.1853  0.2540  0.1249  0.0878  0.3205  0.0000  0.0030  0.0244  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0596  0.0187  0.0397  0.1351  0.1845  0.0669  0.1465  0.1186  0.1229  0.0218  0.0093  0.0297  0.0000   0.0172 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0979  0.2125  0.0898  0.1208  0.3345  0.0328  0.0623  0.0495  0.0000  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0500  0.0296  0.0000  0.0000  0.0000  0.0512  0.0461  0.0763  0.2444  0.1335  0.1058  0.1248  0.0054  0.0000  0.0287  0.0000   0.0180 0.0206  0.0452  0.0206  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0261  0.0053  0.1730  0.2742  0.2064  0.1193  0.0338  0.0633  0.0647  0.0000  0.0000   0.0159 0.0000  0.0000  0.0180  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0216  0.0000  0.0000  0.0000  0.0321  0.0879  0.0651  0.2509  0.2210  0.0791  0.0122  0.0426  0.0794  0.0387  0.0189   0.0506 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0583  0.0559  0.0124  0.0350  0.1122  0.1184  0.2100  0.0422  0.2121  0.1384  0.0051  0.0000   0.0000 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0721  0.0050  0.0706  0.1523  0.0687  0.2586  0.0903  0.0942  0.0645  0.0537  0.0145  0.0127   0.0247 0.0000  0.0182  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0025  0.0000  0.1273  0.2333  0.1096  0.2225  0.0569  0.1483  0.0262  0.0084  0.0440  0.0054   0.0157 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0205  0.0000  0.0000  0.0386  0.0983  0.1957  0.1996  0.0845  0.1283  0.1554  0.0221  0.0184  0.0271   0.0058 0.0000  0.0056  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0238  0.1309  0.1836  0.0885  0.0596  0.0331  0.0243  0.0539  0.0116  0.0000   0.0049 0.1959  0.1900  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0286  0.1052  0.0127  0.0331  0.0453  0.5270  0.0681  0.0175  0.0000  0.1557  0.0000  0.0000   0.0015 0.0054  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0101  0.0333  0.0139  0.0879  0.0998  0.1486  0.0380  0.0292  0.0839  0.0950  0.0012  0.0267   0.0000 0.1008  0.1008  0.1008  0.0302  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0877  0.2994  0.1611  0.1201  0.0673  0.0160  0.0407  0.1560  0.0048   0.0469 0.0000  0.0000  0.0000  0.0000  0.0
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0014  0.0117  0.0286  0.0842  0.0987  0.2034  0.1055  0.0884  0.0975  0.0348   0.1018 0.0353  0.0910  0.0176  0.0000  0.0
##Starting and ending year of mrfss age composition data
2002
2006
##sample sizes of mrfss age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
20.0 13.0 7.0 16.0 8.0
20.0 13.0 7.0 16.0 8.0
##mrfss age comps (year,lengthbin)
0.0000  0.0000  0.0191  0.8552  0.1011  0.0055  0.0000  0.0164  0.0000  0.0027  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.1324  0.5588  0.2500  0.0441  0.0147  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.3148  0.5370  0.1019  0.0463  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.3382  0.3382  0.2415  0.0628  0.0193  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.5000  0.3962  0.0189  0.0472  0.0283  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0094

##################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)= 1.42, 3.62
# zero in slope2 provides logistic selectivity

0.1 #selpar_L50_RVC
0.43 #selpar_slope_RVC
4.0 #selpar_L502_RVC
0.0  #selpar_slope2_RVC

3.65 #selpar_L50_CVT
1.52 #selpar_slope_CVT
1.1 #selpar_L502_CVT
0.4 #selpar_slope2_CVT

2.0 #selpar_L50_cL2
0.8 #selpar_slope_cL2
4.0 #selpar_L502_cL2
0.0 #selpar_slope2_cL2
3.7 #selpar_L50_cL3
2.5 #selpar_slope_cL3
4.0 #selpar_L502_cL3
0.0 #selpar_slope2_cL3

1.42 #selpar_L50_comm02
1.52 #selpar_slope_comm02
1.1 #selpar_L502_comm02
0.4 #selpar_slope2_comm02
3.62 #selpar_L50_comm03
#2.0 #selpar_slope_comm03
#3.0 #selpar_L502_comm03
#0.25 #selpar_slope2_comm03
```

```
1.9 #selpar_L50_HB1
1.5 #selpar_slope_HB1
4.0 #selpar_L502_HB1
0.0 #selpar_slope2_HB1
2.0 #selpar_L50_HB2
3.0 #selpar_slope_HB2
4.0 #selpar_L502_HB2
0.0 #selpar_slope2_HB2
2.7 #selpar_L50_HB3
3.5 #selpar_slope_HB3
4.0 #selpar_L502_HB3
0.0 #selpar_slope2_HB3

0.5 #selpar_Age1_HB_D3

#1.5 #selpar_L50_HB_D3
#3.0 #selpar_slope_HB_D3
#2.2 #selpar_L502_HB_D3
#3.0 #selpar_slope2_HB_D3

3.2 #selpar_L50_MRFSS3
3.5 #selpar_slope_MRFSS3
4.0 #selpar_L502_MRFSS3
0.0 #selpar_slope2_MRFSS3

#################Likelihood Component Weighting###############################################################################
##Weights in objective fcn
1.0     #landings
1.0     #discards
1.0     #length comps
1.0     #age comps
1.0     #RVC index
1.0     #CVT index
1.0     #commercial HAL index
1.0     #HB index
1.0     #MRFSS index
1.0     #S-R residuals
0.0     #constraint on early recruitment deviations
0.0     #constraint on ending recruitment deviations
100.0   #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization)
100.0   #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 rec landings (MRFSS)
1.0

##log catchabilities (initial guesses)
-13.0     #RVC survey
-13.0    #Marmap CVT
-8.0    #commHAL (index in weight)
-13.0    #HB
-13.0    #MRFSS
#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
-1
##annual positive rate of increase on all fishery dependent q's due to technology creep
0.0
# DD q switch: Integer value (choose estimation phase, negative value turns it off)
-1
##density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)
0.0
##SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)
0.128
#Age to begin counting D-D q (should be age near full exploitation)
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
-4.0            #commOther
-5.0            #HB
-3.0            #MRFSS
#Initialization F as a proportion of  first few assessment years (set to 1.0 without evidence otherwise)
1.0
#log mean F's for discards (initial guesses)
-5.0            #commHAL discards
-5.0            #HB discards
-3.0            #MRFSS discards
#Early commercial discard F as a proportion of later period
1.0
#Early HB discard F as a proportion of later period
1.0

#multiplicative adjustment to landings/discards CVs
5.0

#Tuning F (not applied in last phase of optimization)
0.35
#Year for tuning F
2008



##threshold sample sizes for length comps (set to 99999.0 if sel is fixed)
1.0 #RVC
1.0 #CVT
1.0 #HAL
1.0 #cOther
1.0 #HB
```

```
1.0 #HB discards
1.0 #REC

#threshold sample sizes (greater than or equal to) for age comps
1.0 #CVT
1.0 #HAL
1.0 #HB
1.0 #REC
#
#Ageing error matrix (columns are true age 1-20, rows are ages as read for age comps)
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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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_cO2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_cO3  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_MRFSS2  proportion of length comp mass below size limit considered when matching length comp
0.0 #p_lenc_MRFSS3  proportion of length comp mass below size limit considered when matching length comp

0.0 #p_lenc_cL_D2  proportion of length comp mass above size limit considered when matching length comp of discards
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_MRFSS_D2  proportion of length comp mass above size limit considered when matching length comp of discards
0.0 #p_lenc_MRFSS_D3  proportion of length comp mass above size limit considered when matching length comp of discards


999 #end of data file flag
```