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

Sustainable Fisheries Branch, NMFS Beaufort Lab

SEDAR25-RW04

Date Submitted: 26 September 2011

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

Sustainable Fisheries Branch

National Marine Fisheries Service

Southeast Fisheries Science Center

NOAA Beaufort Laboratory

101 Pivers Island Road, Beaufort, NC 28516

1 Overview

The primary model in this assessment was the Beaufort assessment model (BAM), which applies a statistical catch-age formulation. The model was implemented with the AD Model Builder software (ADMB Foundation 2011), and its structure and equations are detailed herein. In essence, a statistical catch-age model simulates a population forward in time while including fishing processes (Quinn and Deriso 1999; Shertzer et al. 2008). Quantities to be estimated are systematically varied until characteristics of the simulated populations match available data on the real population. Statistical catch-age models share many attributes with ADAPT-style tuned and untuned VPAs.

The method of forward projection has a long history in fishery models. It was introduced by Pella and Tomlinson (1969) for fitting production models and then, among many applications, used by Fournier and Archibald (1982), by Deriso et al. (1985) in their CAGEAN model, and by Methot (1989; 2009) in his Stock Synthesis model. The catch-age model of this assessment is similar in structure to the CAGEAN and Stock Synthesis models. Versions of this assessment model have been used in previous SEDAR assessments of reef fishes in the U.S. South Atlantic, such as red porgy, black seabass, snowy grouper, gag grouper, greater amberjack, vermilion snapper, Spanish mackerel, red grouper, and red snapper, as well as in the previous tilefish benchmark (SEDAR 4).

2 Model configuration and equations

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

Stock dynamics In the assessment model, new biomass was acquired through growth and recruitment, while abundance of existing cohorts experienced exponential decay from fishing and natural mortality. The population was assumed closed to immigration and emigration. The model included age classes $1 - 25^+$, where the oldest age class 25^+ allowed for the accumulation of fish (i.e., plus group).

Initialization Initial (1962) abundance at age was estimated in the model as follows. First, the equilibrium age structure was computed for ages 1–25 based on natural and fishing mortality (F), where F was set equal to a value that resulted in the 1962 biomass level equaling 90% of the unfished level. This was based on the assumption by the assessment workshop panel that the stock was lightly exploited prior to the 1960's.

Natural mortality rate The natural mortality rate (M) was assumed constant over time, but decreasing with age. The form of M as a function of age was based on Lorenzen (1996). The Lorenzen (1996) approach inversely relates the natural mortality at age to mean weight at age W_a by the power function $M_a = \alpha W_a^{\beta}$, where α is a scale parameter and β is a shape parameter. Lorenzen (1996) provided point estimates of α and β for oceanic fishes, which were used for this assessment. As in previous SEDAR assessments, the Lorenzen estimates of M_a were rescaled to provide the same fraction of fish surviving from age-1 through the oldest observed age (40 yr) as would occur with constant $M = 0.10$ from the DW. This approach using cumulative mortality is consistent with the findings of Hoenig (1983) and Hewitt and Hoenig (2005).

Growth Mean size at age of the population (total length, TL) was modeled with the von Bertalanffy equation, and weight at age (whole weight, WW) was modeled as a function of total length. Parameters of growth and conversions (TL-WW) were estimated by the DW and were treated as input to the assessment model. The von Bertalanffy parameter estimates from the DW were $L_\infty = 825.1$, $k = 0.189$, and $t_0 = -0.47$. For fitting length composition data, the distribution of size at age was assumed normal with coefficient of variation (CV) estimated by the assessment model. A constant CV, rather than constant standard deviation, was suggested by the size at age data.

Female maturity Females were modeled to be fully mature at age 4 and the proportion mature at ages 1, 2, and 3 were estimated to be 0.1, 0.25, and 0.5 respectively (Table 2.1).

Spawning stock Spawning stock was modeled using mature female gonad weight measured at the time of peak spawning. For tilefish, peak spawning was considered to occur in May. In cases when reliable estimates of fecundity

are unavailable, spawning biomass, and in this case, female gonad weight, is commonly used as a proxy for population fecundity.

Recruitment Expected recruitment of age-1 fish was predicted from spawning stock using the Beverton–Holt spawner-recruit model. Annual variation in recruitment was assumed to occur with lognormal deviations for years 1976–2003 only. The start of recruitment residuals in 1976 was based on examination of a series of different starting years and the start of the age and length composition data that have information on year class strength. The ending year of estimated recruitment residuals (2003) is based on the age at full selection in the fisheries and the last year of age composition data.

Because the age at full selection for the tilefish fisheries generally occurs at age 7 and the last year of composition data in the model is 2010, the assessment panel agreed that recruitment deviations during 2004–2010 could not be reliably estimated.

Landings The model included time series of landings from three fleets: commercial longlines (1962-2010), commercial handlines (1962-2010), and general recreational (1981-2010). An "other" category in the reported landings was distributed by year between handlines and longlines based on the yearly ratio of handline to longline landings.

Landings were modeled with the Baranov catch equation (Baranov 1918) and were fitted in units of weight (1000 lb whole weight). The DW provided observed landings back to the first assessment year (1962) for each fleet except general recreational, because the MRFSS estimates started in 1981.

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

Selectivities Selectivity curves applied to landings and CPUE series were estimated using a parametric approach. This approach applies plausible structure on the shape of the curves, and achieves greater parsimony than occurs with unique parameters for each age. Selectivity of landings from all fleets were modeled as flat-topped, using a two parameter logistic function. Selectivities of the fishery-dependent index was the same as that of the longline fleet. The MARMAP index was also modeled as a flat-topped, two parameter logistic function. However, a selectivity curve was not estimated for the recreational fleet due to low sample sizes and noisy composition data. Instead, the recreational selectivity was assumed to be equal to the commercial handline fishery, since both sectors use vertical hook and line.

Diffuse priors were used for estimating parameters of selectivity functions. These priors assumed normal distributions with CV = 1.0 and were intended to provide only weak information to help the optimization routine during model execution. Priors help by steering estimation away from parameter space with no response in the likelihood surface. Without these diffuse priors, it is possible during the optimization search that a selectivity parameter could become unimportant, for example if its bounds were set too wide and depending on values of other parameters. When this happens, the likelihood gradient with respect to the aimless parameter approaches zero even if the parameter is not at its globally best value. Diffuse priors help avoid that situation.

Indices of abundance The model was fit to two indices of relative abundance: MARMAP longline (binned years between 1985 and 2010) and commercial lines (1993–2010). The MARMAP index values were means of the binned years for both the observed and the predicted series. Predicted indices were conditional on selectivity of the corresponding fleet or survey and were computed from abundance or biomass (as appropriate) at the midpoint of the year.

Catchability In the BAM, catchability scales indices of relative abundance to estimated population abundance at large. Several options for time-varying catchability were implemented in the BAM following recommendations of the 2009 SEDAR procedural workshop on catchability (SEDAR Procedural Guidance 2009). In particular, the BAM allows for density dependence, linear trends, and random walk, as well as time-invariant catchability. Parameters for these models could be estimated or fixed based on *a priori* considerations. For the base model, the AW assumed

time-invariant catchability, following SEDAR 4. For a sensitivity run, however, the AW considered linearly increasing catchability with a slope of 2%, constant after 2003. Choice of the year 2003 was based on recommendations from fishermen regarding when the effects of Global Positioning Systems likely saturated in the southeast U.S. Atlantic (SEDAR 2009). This trend reflects the belief that catchability has generally increased over time as a result of improved technology (SEDAR Procedural Guidance 2009) and as estimated for reef fishes in the Gulf of Mexico (Thorson and Berkson 2010). The value of 2% has been found in other fisheries as well (Zhou et al. 2011). Another sensitivity run applied a random walk to catchability. This is notoriously difficult to estimate, often resulting in just an absorption of noise from the index. The random walk sensitivity run should not be considered a viable model run.

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

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

The model includes the capability for each component of the likelihood to be weighted by user-supplied values (for instance, to give more influence to stronger data sources). For data components, these weights were applied by either adjusting CVs (lognormal components) or adjusting effective sample sizes (multinomial components). In this application to tilefish, CVs of landings (in arithmetic space) were assumed equal to 0.05, to achieve a close fit to these time series yet allowing some imprecision. In practice, the small CVs are a matter of computational convenience, as they help achieve the desired result of close fits to the landings, while avoiding having to solve the Baranov equation iteratively (which is complex when there are multiple fisheries). Weights on other data components (indices, age/length compositions) were adjusted iteratively, starting from initial weights as follows. The CVs of indices were set equal to the values estimated by the DW. Effective sample sizes of the multinomial components were assumed equal to the number of trips sampled annually, rather than the number of fish measured, reflecting the belief that the basic sampling unit occurs at the level of trip. These initial weights were then adjusted until standard deviations of normalized residuals were near 1.0 (SEDAR25-RW04, SEDAR25-RW06). The weight on the commercial longline index was then adjusted upward to a value of 3 (SEDAR25-RW06), in accordance with the principle that abundance data should be given primacy (Francis 2011). A range of weights for the commercial longline index component were considered (ranging from 1.0 to 6.0) before the final 3.0 weight was selected by the AW panel.

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

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Quantity	Symbol	Description or definition
General Definitions		
Index of years	y	$y \in \{1962 \dots 2010\}$
Index of ages	a	$a \in \{1, 2 \dots A\}$, where $A = 25^{+}$
Index of length bins	l	$l \in \{1, 2 \dots 23\}$
Length bins	l'	$l' \in \{340, 370, \dots, 1150\text{mm}\}$, with midpoint of 30mm bin used to match length compositions. Largest 5 length bins treated as a plus group, but retained for weight calculations.
Index of fisheries	f	$f \in \{1, 2, 3\}$ where 1 = commercial longlines, 2 = commercial handlines, 3 = recreational
Index of CPUE	u	$u \in \{1, 2\}$ where 1 = MARMAP longline, 2 = commercial logbook
Conversion factor for whole to gutted weight	b	$b = 1.05893$
Input Data		
Observed length compositions	$p^{\lambda}_{(f,u),l,y}$	Proportional contribution of length bin l in year y to fishery f (landings) or index u
Observed age compositions	$p^{\alpha}_{(f,u),a,y}$	Proportional contribution of age class a in year y to fishery f or index u.
Ageing error matrix	$\mathcal{E}$	Estimated from multiple readers ageing the same otoliths.
Length comp. sample sizes	$n^{\lambda}_{(f,u),y}$	Effective number of length samples collected in year y from fishery f or index u
Age comp. sample sizes	$n^{\alpha}_{(f,u),y}$	Effective number of age samples collected in year y from fishery f or index u
Observed landings	$L_{f,y}$	Reported landings in year y from fishery f. Commercial L in 1000 lb gutted weight, and recreational L in 1000 lb whole weight.
CVs of landings	$c^{L}_{f,y}$	Assumed 0.05 in arithmetic space
Observed abundance indices	$U_{u,y}$	$u = 1$, MARMAP longline (numbers), $y \in \{1985, 1998, 2002, 2006, 2010\}$ $u = 2$, Commercial longline logbook (weight), $y \in \{1981 \dots 2010\}$
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1, 2\}$ as above. Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean
Natural mortality rate	M_a	Function of weight at age (w_a): $M_a = \alpha w_a^{\beta}$, with estimates of α and β from Lorenzen (1996). Lorenzen M_a then rescaled based on Hoenig estimate.
Population Model		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages
Proportion females mature at age	m_a	Logistic increase with age; assumed constant across years

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Spawning date	t_{spawn}	Fraction denoting the proportional time of year when spawning occurs. Set to 0.42 for tilefish by assuming peak spawning occurs in the end of May.
Mean length at age	l_a	Total length (midyear); $l_a = L_\infty(1 - \exp[-K(a - t_0 + 0.5)])$ where K, L_∞, and t_0 are parameters estimated by the DW
CV of l_a	$\widehat{c}_a^{\lambda}$	Estimated coefficient of variation of growth, assumed constant across ages
SD of l_a	σ_a^{λ}	Standard deviation of growth, assumed constant across ages.
Age–length conversion of population	$\psi_{a,l}^{u}$	$\psi_{a,l}^{u} = \frac{1}{\sqrt{2\pi}(\sigma_a^{\lambda})} \frac{\exp\left[-\left(l_l' - l_a\right)^2\right]}{\left(2(\sigma_a^{\lambda})^2\right)}$, the Gaussian density function. Matrix ψ^u is rescaled to sum to one within ages, with the largest size a plus group. This matrix is constant across years and is used only to match length comps of fishery independent indices.
Age–length conversion of landings	$\psi_{f,a,l,y}^{L}$	$\psi_{f,a,l,y}^{L} = \psi_{a,l}^{u}$
Mean length at age of landings	$\xi_{(f),a,y}^{L}$	Mean length at age from $\psi_{f,a,y}^{L}$ for landings.
Individual weight at age of population	w_a	Computed from length at age by $w_a = \theta_1 l_a^{\theta_2}$ where θ_1 and θ_2 are parameters from the DW
Gonad weight at age of individuals	g_a	Computed from weight at age by $g_a = \varpi_1 w_a^{\varpi_2}$ where ϖ_1 and ϖ_2 are parameters from the DW
Individual weight at age of landings	$w_{(f),a,y}^{L}$	Computed from length at age by $w_{(f),a,y}^{L} = \theta_1(\xi_{(f),a,y}^{L})^{\theta_2}$
Fishery and index selectivities	$s_{(f,u),a}$	$s_{(f,u),a} = \frac{1}{1+\exp\left[-\widehat{\eta}_{(f,u)}\left(a - \widehat{\alpha}_{(f,u)}\right)\right]}$ where $\widehat{\eta}_{(f,u)}$ and $\widehat{\alpha}_{(f,u)}$ are estimated parameters. Not all parameters were estimated for each fishery or index; some parameters were fixed as described in the text. For instance, the selectivity of the commercial logbook index was assumed equal to that of commercial longlines. Commercial line selectivity was used as a proxy for the recreational fleet.
Fishing mortality rate of landings	$F_{f,a,y}$	$F_{f,a,y} = s_{f,a,y}\widehat{F}_{f,y}$ where $\widehat{F}_{f,y}$ is an estimated fully selected fishing mortality rate by fishery
Total fishing mortality rate	$F_{a,y}$	$F_{a,y} = \sum_f F_{f,a,y}$
Total mortality rate	$Z_{a,y}$	$Z_{a,y} = M_a + F_{a,y}$
Apical F	F_y	$F_y = \max(F_{a,y})$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Abundance at age	$N_{a,y}$	$N_{1,1962} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}\phi_{init}-0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)\phi_{init}}$ $\widehat{N}_{1+,1962}$ equilibrium conditions expected given assumptions about initial fishing mortality (described below) $N_{0,y+1} = \frac{0.8\widehat{R}_0\widehat{h}S_y}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_y}\exp(\widehat{R}_{y+1})$for $y \geq 1962$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (0 \dots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ $\widehat{R}_0$ (asymptotic maximum recruitment) is an estimated parameter of the spawner-recruit curve, and $\widehat{R}_y$ are estimated annual recruitment deviations in log space. The bias correction is $\varsigma = \exp(\widehat{\sigma_R}^2/2)$, where $\widehat{\sigma_R}^2$ is the estimated variance of recruitment deviations. In the SEDAR-25 baserun, $h = 0.84$ was a fixed parameter. Quantities ϕ_0, ϕ_{init}, and S_y are described below.
Abundance at age (mid-year)	$N'_{a,y}$	Used to match indices of abundance $N'_{a,y} = N_{a,y}\exp(-Z_{a,y}/2)$
Abundance at age at time of spawning	$N''_{a,y}$	Assumed in May to correspond with peak spawning $N''_{a,y} = \exp(-t_{\text{spawn}}Z_{a,y})N_{a,y}$
Unfished abundance at age per recruit at time of spawning	NPR_a	$NPR_0 = 1 \times \exp(-t_{\text{spawn}}M_0)$ $NPR_{a+1} = NPR_a\exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (0 \dots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_A t_{\text{spawn}})]}{1-\exp(-M_A)}$
Initial abundance at age per recruit at time of spawning	NPR_a^{init}	Same calculations as for NPR_a, but including fishing mortality (see Z^{init} below).
Unfished spawning biomass per recruit	ϕ_0	$\phi_0 = \sum_{a=0}^{A} NPR_a\rho_a m_a g_a$ In units of mature female gonad weight.
Initial spawning biomass per recruit	ϕ_{init}	$\phi_{init} = \sum_{a=0}^{A} NPR_a^{init}\rho_a m_a g_a$ In units of mature female gonad weight.
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y}\rho_a m_a g_a$ Also referred to as spawning biomass in units of total mature female gonad weight.
Initialization mortality at age	Z_a^{init}	$Z_a^{init} = M_a + s_a^{init}F^{init}$ where F^{init} is an initialization F assumed to be that required to match a 10% depletion from virgin conditions in 1962. s_a^{init} is set to the commercial line selectivity. In the SEDAR-25 baserun, $F^{init} = 0.01$ was fixed.
Initial equilibrium abundance at age	N_a^{equil}	Equilibrium age structure given Z_a^{init}
Population biomass	B_y	$B_y = \sum_a N_{a,y}w_a$
Landings at age in numbers	$L'_{f,a,y}$	$L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}}N_{a,y}[1-\exp(-Z_{a,y})]$
Landings at age in whole weight	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y}L'_{f,a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Landings at age in gutted weight	$L'''_{f,a,y}$	$L'''_{f,a,y} = bL''_{f,a,y}$
Index catchability	$q_{u,y}$	$q_{u,1993} = \widehat{q}^0_u f(\text{density})$ $q_{u,y+1} = q_{u,y} f_y(\text{trend}) f_y(\text{random}) f_y(\text{density})$ for $y \geq 1993$ Here, $f_y(\text{density}) = (B'_0)^{\widehat{\psi}}(B'_y)^{-\widehat{\psi}}$, where $\widehat{\psi}$ is a parameter to be estimated, $B'_y = \sum_{a=a'}^{A} B_{a,y}$ is annual biomass above some threshold age a', and B'_0 is virgin biomass for ages a' and greater. In practice, a' should be set high enough to give a reasonable summary of exploitable biomass. The function $f(\text{trend})$ provides a model for linear trend (slope of β_q) in catchability from the start of the index until 2003, where technology effects were thought to saturate (see SEDAR 19 DW report). For example, for an index that starts in 1993, $f_y(\text{trend})$ follows, $f_y(\text{trend}) = \begin{cases} 1.0 & :y = 1993 \\ f_{y-1}(\text{trend}) * (\text{y} - 1993)\beta_{\text{q}} & :1993 < y \leq 2003 \\ f_{2003}(\text{trend}) & :2003 < y \end{cases}$ Finally, $f_y(\text{random}) = \exp(\epsilon_{u,y})$ are lognormal catchability deviations which allow for a random walk in catchability when penalties are placed on the $\epsilon_{u,y}$ (see "Objective Function"). In practice, the catchability function $f_y(\text{trend})$ was used as described for the SEDAR-25 tilefish assessment. Density dependence and random walks were not applied in the baserun.
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \begin{cases} \sum_a L'''_{f,a,y} & :f = 1,2 \\ \sum_a L''_{f,a,y} & :f = 3 \end{cases}$
Predicted length compositions of fishery independent data	$\breve{p}^{\lambda}_{u,l,y}$	$\breve{p}^{\lambda}_{u,l,y} = \frac{\sum_a \psi_{a,l} s_{u,a,y} N'_{a,y}}{\sum_a s_{u,a,y} N'_{a,y}}$
Predicted length compositions of landings	$\breve{p}^{\lambda}_{f,l,y}$	$\breve{p}^{\lambda}_{f,l,y} = \frac{\sum_a \psi^L_{f,a,l,y} L'_{f,a,y}}{\sum_a L'_{f,a,y}}$
Predicted age compositions	$\breve{p}^{\alpha}_{(f,u),a,y}$	$\breve{p}^{\alpha}_{(f,u),a,y} = \frac{L'_{(f,u),a,y}}{\sum_a L'_{(f,u),a,y}}$
Predicted CPUE	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \begin{cases} \widehat{q}_{u,y} \sum_a N'_{a,y'} s_{u,a} & : \quad u = 1 \\ \widehat{q}_{u,y} \sum_a w^L_{u,a,y} N'_{a,y} s_{u,a} & : \quad u = 2 \end{cases}$ where y' corresponds to the years used as means of the binned index. $s_{u,a}$ is the selectivity of the relevant fishery in the year corresponding to y.

Table 2.1. (continued)

Quantity	Symbol	Description or definition
Objective Function		
Multinomial length compositions	Λ_1	$\Lambda_1 = -\sum_{f,u}\sum_{y}\left[\omega^{\lambda}_{(f,u)} n^{\lambda}_{(f,u),y} \sum_{l}(p^{\lambda}_{(f,u),l,y}+x)\log\left(\frac{(\breve{p}^{\lambda}_{(f,u),l,y}+x)}{(p^{\lambda}_{(f,u),l,y}+x)}\right)\right]$ where $\omega^{\lambda}_{(f,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term. Bins are 30 mm wide.
Multinomial age compositions	Λ_2	$\Lambda_2 = -\sum_{f,u}\sum_{y}\left[\omega^{\alpha}_{(f,u)} n^{\alpha}_{(f,u),y} \sum_{a}(p^{\alpha}_{(f,u),a,y}+x)\log\left(\frac{(\breve{p}^{\alpha}_{(f,u),a,y}+x)}{(p^{\alpha}_{(f,u),a,y}+x)}\right)\right]$ where $\omega^{\alpha}_{(f,u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term.
Lognormal landings	Λ_3	$\Lambda_3 = \sum_{f}\sum_{y}\frac{\left[\log\left((L_{f,y}+x)\big/(\breve{L}_{f,y}+x)\right)\right]^2}{2(\sigma^L_{f,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^L_{f,y} = \sqrt{\log(1+(c^L_{f,y}/\omega^L_f)^2)}$, with $\omega^L_f = 1$ a preset weight.
Lognormal CPUE	Λ_4	$\Lambda_4 = \sum_{u}\sum_{y}\frac{\left[\log\left((U_{u,y}+x)\big/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma^U_{u,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma^U_{u,y} = \sqrt{\log(1+(c^U_{u,y}/\omega^U_u)^2)}$, with ω^U_u a preset weight.
Lognormal recruitment deviations	Λ_5	$\Lambda_5 = \omega_5\left[\frac{[R_{1976}+(\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y>1976}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where R_y are recruitment deviations in log space, n is the number of years, $\omega_5 = 1$ is a preset weight, $\widehat{\varrho}$ is the first-order autocorrelation, and $\widehat{\sigma}^2_R$ is the estimated recruitment variance ($\varrho = 0$ in the SEDAR-25 base run).
Additional constraint on early recruitment deviations	Λ_6	$\Lambda_6 = \omega_6\left[\frac{[R_{1976}+(\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y=1977}^{Y_1}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where Y_1 is the last year to apply this additional penalty and ω_6 is a preset weight, with ω_6=0.0 for the SEDAR-25 tilefish base run.
Additional constraint on final recruitment deviations	Λ_7	$\Lambda_7 = \omega_7\left[\sum_{y=Y_2}^{Y}\frac{[(R_y-\widehat{\varrho}R_{y-1})+(\widehat{\sigma}^2_R\big/2)]^2}{2\widehat{\sigma}^2_R} + n\log(\sigma_R)\right]$ where Y_2 is the first year to apply this additional penalty, Y is the terminal year, and ω_7 is a preset weight, with ω_7=0.0 for the SEDAR-25 tilefish base run.
Penalty on random walk on catchability	Λ_8	$\Lambda_8 = \omega_8\sum_{u}\sum_{y}\frac{\epsilon^2_{u,y}}{2(\sigma^q_u)^2}$ where ω_8 is a preset weight and σ^q_u is a control variable input by the user defining the standard deviation of the random walk process. As σ^q_u increases, one essentially estimates each deviation as a free parameter, while values close to zero allow little variation in annual catchability. A random walk on catchability was not used for the SEDAR-25 tilefish baserun, thus ω_8=0.0.
Penalty on initial age structure	Λ_9	$\Lambda_9 = N^{eq}_a$ where N^{eq}_a is the equilibrium age structure given the initial F, as defined previously.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Prior distributions and penalties	Λ_{10}	is the sum of penalty terms used to implement prior distributions on several parameters. Normal priors were applied to $\widehat{\sigma}_R$, $\widehat{c}_a^{\lambda}$, $\widehat{\eta}_{(f,u)}$, and $\widehat{\alpha}_{(f,u)}$. Normal distributions required a value to describe variance. Empirical estimates of standard deviation were available and therefore used for c_a^{λ} and σ_R. All other normal priors assumed CV=0.5 (i.e., diffuse priors).
Apical F penalty	Λ_{11}	$\Lambda_{11} = \begin{cases} 0 & :F_{apex} < 1 \\ \omega_{11} \times \exp^{\sqrt{(F_{apex}-1)}} - 1 & :F_{apex} > 1 \end{cases}$ where $\omega_{11} = 10$.
Total objective function	Λ	$\Lambda = \sum_{i=1}^{11} \Lambda_i$ Objective function minimized by the assessment model

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

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR 25 Assessment: Tilefish, June 2011
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

DATA_SECTION

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

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

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

// ending year for first selectivity period
init_int endyr_period1;

//Total number of ages
init_int nages;

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

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

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

//Vector of lengths for length bins (mm)(midpoint) and bins used in computation of plus group
init_ivector lenbins(1,nlenbins);
init_ivector lenbins_plus(1,nlenbins_plus);
int nlenbins_all;    //largest size class used to compute average lengths and weights
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   nlenbins_all=nlenbins+nlenbins_plus;
 END_CALCS

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

//###################Commercial Longline 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 gutted 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

// Length Compositions (3 cm bins)
init_int nyr_cL_lenc;
init_ivector yrs_cL_lenc(1,nyr_cL_lenc);
init_vector nsamp_cL_lenc(1,nyr_cL_lenc);
init_matrix obs_cL_lenc(1,nyr_cL_lenc,1,nlenbins);
// Age Compositions
init_int nyr_cL_agec;
init_ivector yrs_cL_agec(1,nyr_cL_agec);
init_vector nsamp_cL_agec(1,nyr_cL_agec);
init_matrix obs_cL_agec(1,nyr_cL_agec,1,nages);
```

```
//************************************************************************
//##Commercial handline fishery
// Landings (1000 lb gutted weight)
init_int styr_cH_L;
init_int endyr_cH_L;
init_vector obs_cH_L(styr_cH_L,endyr_cH_L);
init_vector cH_L_cv(styr_cH_L,endyr_cH_L);    //vector of CV of landings by year
// Length Compositions (3 cm bins, data from handline)
init_int nyr_cH_lenc;
init_ivector yrs_cH_lenc(1,nyr_cH_lenc);
init_vector nsamp_cH_lenc(1,nyr_cH_lenc);
init_matrix obs_cH_lenc(1,nyr_cH_lenc,1,nlenbins);
// Age Compositions (data from handline)
init_int nyr_cH_agec;
init_ivector yrs_cH_agec(1,nyr_cH_agec);
init_vector nsamp_cH_agec(1,nyr_cH_agec);
init_matrix obs_cH_agec(1,nyr_cH_agec,1,nages);

//************************************************************************
//################################Recreational (all sectors) fishery ######################################
// Landings (1000 fish)
init_int styr_rA_L;
init_int endyr_rA_L;
init_vector obs_rA_L(styr_rA_L,endyr_rA_L);
init_vector rA_L_cv(styr_rA_L,endyr_rA_L);
// Length Compositions (3 cm bins) of landings
init_int nyr_rA_lenc;
init_ivector yrs_rA_lenc(1,nyr_rA_lenc);
init_vector nsamp_rA_lenc(1,nyr_rA_lenc);
init_matrix obs_rA_lenc(1,nyr_rA_lenc,1,nlenbins);

//###################### MARMAP Longline data ############################
//CPUE
init_int nyr_mm_cpue;
init_ivector yrs_mm_cpue(1,nyr_mm_cpue);  //middle year in group of years to be combined
init_vector obs_mm_cpue(1,nyr_mm_cpue);//Observed CPUE
init_vector mm_cpue_cv(1,nyr_mm_cpue); //CV of cpue
// Length Compositions (3 cm bins)
//init_int nyr_mm_lenc;
//init_ivector yrs_mm_lenc(1,nyr_mm_lenc);  //middle year in group of years to be combined
//init_vector nsamp_mm_lenc(1,nyr_mm_lenc);
//init_matrix obs_mm_lenc(1,nyr_mm_lenc,1,nlenbins);
// Age Compositions (data from MARMAP)
init_int nyr_mm_agec;
init_ivector yrs_mm_agec(1,nyr_mm_agec);  //middle year in group of years to be combined
init_vector nsamp_mm_agec(1,nyr_mm_agec);
init_matrix obs_mm_agec(1,nyr_mm_agec,1,nages);

//#################Parameter values and initial guesses ################################
//************************************************************************
// Von Bert parameters in TL mm all fish
init_number set_Linf;
init_number set_K;
init_number set_t0;
// Von Bert parameters in TL mm females only
init_number set_Linf_f;
init_number set_K_f;
init_number set_t0_f;
//Standard erros of von bert params all fish
init_number set_Linf_se;
init_number set_K_se;
init_number set_t0_se;
//CV of length at age and its standard error all fish
init_number set_len_cv;
init_number set_len_cv_se;
//TL(mm)-weight(whole weight in mt) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;
//weight(whole weight)-gonad weight (units=g) relationship: GW=a+b*W
init_number gwgtpar_a;
init_number gwgtpar_b;
//gutted weight to whole weight conversion factor
init_number gut2whole;
//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age

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

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

//Initial guesses or fixed values of estimated selectivity parameters

init_number set_selpar_L50_cL;
init_number set_selpar_slope_cL;
//init_number set_selpar_L502_cL;
//init_number set_selpar_slope2_cL;
//init_number set_selpar_min_cL;
//init_number set_selpar_afull_cL;
//init_number set_selpar_peak_cL;
//init_number set_selpar_top_cL;
//init_number set_selpar_ascwid_cL;
//init_number set_selpar_decwid_cL;
```

```
//init_number set_selpar_init_cL;
//init_number set_selpar_final_cL;

//init_number set_selpar_L50_cL2;
//init_number set_selpar_slope_cL2;

init_number set_selpar_L50_cH;
init_number set_selpar_slope_cH;
//init_number set_selpar_L502_cH;
//init_number set_selpar_slope2_cH;
//init_number set_selpar_min_cH;
//init_number set_selpar_afull_cH;
//init_number set_selpar_peak_cH;
//init_number set_selpar_top_cH;
//init_number set_selpar_ascwid_cH;
//init_number set_selpar_decwid_cH;
//init_number set_selpar_init_cH;
//init_number set_selpar_final_cH;

init_number set_selpar_L50_rA;
init_number set_selpar_slope_rA;
//init_number set_selpar_L502_rA;
//init_number set_selpar_slope2_rA;
//init_number set_selpar_min_rA;
//init_number set_selpar_afull_rA;
//init_number set_selpar_peak_rA;
//init_number set_selpar_top_rA;
//init_number set_selpar_ascwid_rA;
//init_number set_selpar_decwid_rA;
//init_number set_selpar_init_rA;
//init_number set_selpar_final_rA;

init_number set_selpar_L50_mm;
init_number set_selpar_slope_mm;
//init_number set_selpar_L502_mm;
//init_number set_selpar_slope2_mm;
//init_number set_selpar_min_mm;
//init_number set_selpar_afull_mm;
//init_number set_selpar_peak_mm;
//init_number set_selpar_top_mm;
//init_number set_selpar_ascwid_mm;
//init_number set_selpar_decwid_mm;
//init_number set_selpar_init_mm;
//init_number set_selpar_final_mm;

//--weights for likelihood components-----------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_lc_cL;
init_number set_w_lc_cH;
init_number set_w_lc_rA;
//init_number set_w_lc_mm;
init_number set_w_ac_cL;
init_number set_w_ac_cH;
init_number set_w_ac_mm;
init_number set_w_I_mm;
init_number set_w_I_cL;
init_number set_w_rec;             //for fitting S-R curve
init_number set_w_rec_early;       //additional constraint on early years recruitment
init_number set_w_rec_end;         //additional constraint on ending years recruitment
init_number set_w_fullF;           //penalty for any Fapex>3(removed in final phase of optimization)
init_number set_w_Ftune;           //weight applied to tuning F (removed in final phase of optimization)
//init_number set_w_cvlen_dev;         //penalty on cv deviations at age
//init_number set_w_cvlen_diff;       //penalty on first difference of cv deviations at age

//////--index catchability-------------------------------------------------------------------------------------------------------
init_number set_logq_mm;     //catchability coefficient (log) for MARMAP longline CPUE index
init_number set_logq_cL;      //catchability coefficient (log) for commercial logbook CPUE index

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

////--F's--------------------------------
init_number set_log_avg_F_cL;
init_number set_log_avg_F_cH;
init_number set_log_avg_F_rA;
init_number set_F_init;        //initial F, scaled by F_init_ratio

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

//threshold sample sizes for length comps
init_number minSS_cL_lenc;
init_number minSS_cH_lenc;
init_number minSS_rA_lenc;
//init_number minSS_mm_lenc;

//threshold sample sizes for age comps
init_number minSS_cL_agec;
init_number minSS_cH_agec;
init_number minSS_mm_agec;

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

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

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

init_number end_of_data_file;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   if(end_of_data_file!=999)
   {
     for(iyear=1; iyear<=10; iyear++)
     {
       cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
       cout << "" <<endl;
     }
   }
   else
   {
    cout << "Data File read correctly" << endl;
   }
 END_CALCS


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

  //init_bounded_number Linf(500,1100,2);
  //init_bounded_number K(0.05,0.5,2);
  //init_bounded_number t0(-1.0,0.0,2);
  number Linf;
  number K;
  number t0;
  number Linf_f;
  number K_f;
  number t0_f;
  vector meanlen_TL(1,nages);   //mean Total length (mm) at age all fish
  vector meanlen_TL_f(1,nages);   //mean Total length (mm) at age females only
  vector wgt_g(1,nages);        //whole wgt in g
  vector wgt_kg(1,nages);       //whole wgt in kg
  vector wgt_mt(1,nages);       //whole wgt in mt
  vector wgt_klb(1,nages);      //whole wgt in 1000 lb
  vector wgt_lb(1,nages);       //whole wgt in lb
  vector wgt_g_f(1,nages);        //whole wgt in g
  vector wgt_kg_f(1,nages);       //whole wgt in kg
  vector wgt_mt_f(1,nages);       //whole wgt in mt
  vector wgt_klb_f(1,nages);      //whole wgt in 1000 lb
  vector wgt_lb_f(1,nages);       //whole wgt in lb
  vector gonad_wgt_mt(1,nages); //gonad wgt in mt

  matrix len_cL_mm(styr,endyr,1,nages);       //mean length at age of cL landings in mm (may differ from popn mean)
  matrix gutwgt_cL_klb(styr,endyr,1,nages);      //whole wgt of cL landings in 1000 lb
  matrix len_cH_mm(styr,endyr,1,nages);       //mean length at age of cH landings in mm (may differ from popn mean)
  matrix gutwgt_cH_klb(styr,endyr,1,nages);      //whole wgt of cH landings in 1000 lb
  matrix len_rA_mm(styr,endyr,1,nages);       //mean length at age of recreational landings in mm (may differ from popn mean)
  matrix wgt_rA_klb(styr,endyr,1,nages);      //whole wgt of recreational landings in 1000 lb
  matrix len_mm_mm(styr,endyr,1,nages);      //mean length at age of MARMAP landings in mm (may differ from popn mean)
  matrix wgt_mm_klb(styr,endyr,1,nages);     //whole wgt of MARMAP landings in 1000 lb

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

  //matrices below are used to match length comps
  matrix lenprob_cL(1,nages,1,nlenbins);     //distn of size at age in cL
  matrix lenprob_cH(1,nages,1,nlenbins);     //distn of size at age in cH
  matrix lenprob_rA(1,nages,1,nlenbins);    //distn of size at age in rA
  //matrix lenprob_mm(1,nages,1,nlenbins);    //distn of size at age in mm

  //matrices below pertain to the popn at large, used to compute mean weights
  matrix lenprob_cL_all(1,nages,1,nlenbins_all);     //distn of size at age in cL
  matrix lenprob_cH_all(1,nages,1,nlenbins_all);     //distn of size at age in cH
  matrix lenprob_rA_all(1,nages,1,nlenbins_all);    //distn of size at age in rA
  //matrix lenprob_mm_all(1,nages,1,nlenbins_all);    //distn of size at age in mm

  //set min and max equal for constant sd or cv
  init_bounded_number len_cv_val(0.05,0.5,4);
//  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
//  number log_len_cv
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

////----Predicted length compositions
  matrix pred_cL_lenc(1,nyr_cL_lenc,1,nlenbins);
  matrix pred_cH_lenc(1,nyr_cH_lenc,1,nlenbins);
  matrix pred_rA_lenc(1,nyr_rA_lenc,1,nlenbins);
  //matrix pred_mm_lenc(1,nyr_mm_lenc,1,nlenbins);
////----Predicted age compositions
  matrix pred_cL_agec(1,nyr_cL_agec,1,nages);
  matrix ErrorFree_cL_agec(1,nyr_cL_agec,1,nages);
  matrix pred_cH_agec(1,nyr_cH_agec,1,nages);
```

```
  matrix ErrorFree_cH_agec(1,nyr_cH_agec,1,nages);
  matrix pred_mm_agec(1,nyr_mm_agec,1,nages);
  matrix ErrorFree_mm_agec(1,nyr_mm_agec,1,nages);

//effective sample size applied in multinomial distributions
  vector nsamp_cL_lenc_allyr(styr,endyr);
  vector nsamp_cH_lenc_allyr(styr,endyr);
  vector nsamp_rA_lenc_allyr(styr,endyr);
  //vector nsamp_mm_lenc_allyr(styr,endyr);
  vector nsamp_cL_agec_allyr(styr,endyr);
  vector nsamp_cH_agec_allyr(styr,endyr);
  vector nsamp_mm_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_cL_lenc_allyr_out(styr,endyr);
  vector neff_cH_lenc_allyr_out(styr,endyr);
  vector neff_rA_lenc_allyr_out(styr,endyr);
  //vector neff_mm_lenc_allyr_out(styr,endyr);
  vector neff_cL_agec_allyr_out(styr,endyr);
  vector neff_cH_agec_allyr_out(styr,endyr);
  vector neff_mm_agec_allyr_out(styr,endyr);

//-----Population---------------------------------------------------------------------------------
  matrix N(styr,endyr+1,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and cpue
  matrix N_spawn(styr,endyr,1,nages);       //Population numbers by year and age at peaking spawning: used for SSB
  //init_bounded_vector log_Nage_dev(2,nages,-5,3,1); //log deviations on initial abundance at age
  vector log_Nage_dev(2,nages);
  vector log_Nage_dev_output(1,nages);      //used in output. equals zero for first age
  matrix B(styr,endyr+1,1,nages);           //Population biomass by year and age at start of yr
  vector totB(styr,endyr+1);                //Total biomass by year
  vector totN(styr,endyr+1);                //Total abundance by year
  vector SSB(styr,endyr);                   //Total spawning biomass by year (total mature female gonad weight)
  vector MatFemB(styr,endyr);               //Total spawning biomass by year (total mature female biomass)
  vector rec(styr,endyr+1);                 //Recruits by year
  vector prop_f(1,nages);                   //Proportion female by age
  vector maturity_f(1,nages);               //Proportion of female mature at age
  vector reprod(1,nages);                   //vector used to compute spawning biomass (total mature female gonad weight)
  vector reprod2(1,nages);                  //vector used to compute mature female biomass

////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(8,18,1);        //log(virgin Recruitment)
  //number log_R0;
  number R0;                                  //virgin recruitment
  init_bounded_number steep(0.21,0.991,-3); //steepness
  //number steep;  //uncomment to fix steepness, comment line directly above
  init_bounded_number rec_sigma(0.2,1.2,4);  //sd recruitment residuals
  number rec_sigma_sq;                        //square of rec_sigma
  number rec_logL_add;                        //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr_rec_dev,-5,5,3);  //log recruitment deviations
  //init_bounded_vector log_rec_historic_dev(styr+1,styr_rec_dev-1,-5,5,3); //log recruitment deviations early period
  //vector log_rec_dev(styr_rec_dev,endyr);
  vector log_rec_dev_output(styr,endyr+1);          //used in output. equals zero except for yrs in log_rec_dev
  //vector log_rec_historic_dev_output(styr,endyr+1);  //used in output. equals zero except for yrs in log_rec_dev

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

  number BiasCor;                             //Bias correction in equilibrium recruits
  init_bounded_number R_autocorr(-1.0,1.0,-4);  //autocorrelation in SR
  number S0;                                  //equal to spr_F0*R0 = virgin SSB
  number B0;                                  //equal to bpr_F0*R0 = virgin B
  number R1;                                  //Recruits in styr
  number R_virgin;                            //unfished recruitment with bias correction
  vector SdS0(styr,endyr);                    //SSB / virgin SSB


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

//Commercial longline-----------------------------------------------
  matrix sel_cL(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cL(1.0,10.0,2);
  init_bounded_number selpar_slope_cL(0.5,12.0,2);
  //init_bounded_number selpar_L502_cL(4.0,15.0,-3);
  //init_bounded_number selpar_slope2_cL(0.1,5.0,-3);
  //init_bounded_number selpar_min_cL(0.0,1.0,-3);
  //init_bounded_number selpar_afull_cL(4.0,8.0,-3);  //location in age vector (=age only if age1=1) for joining ascending and descending curves
  //init_bounded_number selpar_peak_cL(0.0,19.8,2);
  //init_bounded_number selpar_top_cL(-5,3,4);
  //init_bounded_number selpar_ascwid_cL(-4,12,4);
  //init_bounded_number selpar_decwid_cL(-2,6,-4);
  //init_bounded_number selpar_init_cL(-15,5,-4);
  //init_bounded_number selpar_final_cL(-5,5,-4);

  //init_bounded_number selpar_L50_cL2(4.0,12.0,2);
  //init_bounded_number selpar_slope_cL2(0.5,12.0,2);

//Commercial handline-----------------------------------------------
  matrix sel_cH(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cH(1.0,10.0,2);
  init_bounded_number selpar_slope_cH(0.5,12.0,2);
  //init_bounded_number selpar_L502_cH(2.0,10.0,-3);
  //init_bounded_number selpar_slope2_cH(0.1,5.0,-3);
  //init_bounded_number selpar_min_cH(0.0,1.0,-3);
  //init_bounded_number selpar_afull_cH(4.0,8.0,-3); //location in age vector (=age only if age1=1) for joining ascending and descending curves
  //init_bounded_number selpar_peak_cH(0.0,19.8,2);
  //init_bounded_number selpar_top_cH(-5,3,2);
  //init_bounded_number selpar_ascwid_cH(-4,12,4);
  //init_bounded_number selpar_decwid_cH(-2,6,-4);
  //init_bounded_number selpar_init_cH(-15,5,-4);
  //init_bounded_number selpar_final_cH(-5,5,-4);
```

```
//Recreational-------------------------------------------------
  matrix sel_rA(styr,endyr,1,nages);
  init_bounded_number selpar_L50_rA(1.0,10.0,-2);
  init_bounded_number selpar_slope_rA(0.5,12.0,-2);
  //init_bounded_number selpar_L502_rA(2.0,10.0,-3);
  //init_bounded_number selpar_slope2_rA(0.1,5.0,-3);
  //init_bounded_number selpar_min_rA(0.0,1.0,-3);
  //init_bounded_number selpar_afull_rA(4.0,8.0,-3); //location in age vector (=age only if age1=1) for joining ascending and descending curves
  //init_bounded_number selpar_peak_rA(0.0,19.8,2);
  //init_bounded_number selpar_top_rA(-5,3,2);
  //init_bounded_number selpar_ascwid_rA(-4,12,4);
  //init_bounded_number selpar_decwid_rA(-2,6,-4);
  //init_bounded_number selpar_init_rA(-15,5,-4);
  //init_bounded_number selpar_final_rA(-5,5,-4);

////MARMAP selectivity-------------------------------------------------
  matrix sel_mm(styr,endyr,1,nages);
  init_bounded_number selpar_L50_mm(1.0,10.0,2);
  init_bounded_number selpar_slope_mm(0.5,12.0,2);
  //init_bounded_number selpar_L502_mm(2.5,15.0,-3);
  //init_bounded_number selpar_slope2_mm(0.1,12.0,-3);
  //init_bounded_number selpar_min_mm(0.0,1.0,-3);
  //init_bounded_number selpar_afull_mm(4.0,8.0,-3); //location in age vector (=age only if age1=1) for joining ascending and descending curves
  //init_bounded_number selpar_peak_mm(0.0,19.8,2);
  //init_bounded_number selpar_top_mm(-5,3,2);
  //init_bounded_number selpar_ascwid_mm(-4,12,4);
  //init_bounded_number selpar_decwid_mm(-2,6,-4);
  //init_bounded_number selpar_init_mm(-15,5,-4);
  //init_bounded_number selpar_final_mm(-5,5,-4);

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

//-------------------------------------------------------------------------------------------------------------------------------------------------
//-------CPUE Predictions-------------------------------
  vector pred_mm_cpue(1,nyr_mm_cpue);          //predicted MARMAP U
  matrix N_mm(1,nyr_mm_cpue,1,nages);          //used to compute MARMAP CPUE index
  vector pred_mm_cpue_allyr(styr,endyr);  // used for graphing purposes, fills in consec years
  vector obs_mm_cpue_allyr(styr,endyr);   // used for graphing purposes, fills in consec years
  vector mm_cpue_cv_allyr(styr,endyr);    // used for graphing purposes, fills in consec years
  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

//---Catchability (CPUE q's)-------------------------------------------------------------
  init_bounded_number log_q_cL(-14,-4,1);
  init_bounded_number log_q_mm(-14,-4,1);
  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)

  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);

  vector q_cL(styr_cL_cpue,endyr_cL_cpue);
  number q_mm;

//-------------------------------------------------------------------------------------------------------------------------------------------------
//---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_cH_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cH_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cH_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cH_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages

  matrix L_rA_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_rA_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_rA_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_rA_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages
  vector obs_rA_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicative bias

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

////---MSY calcs-------------------------------------------------------------------------
  number F_cL_prop;       //proportion of F_sum attributable to hal, last X=selpar_n_yrs_wgted yrs, used for avg body weights
  number F_cH_prop;       //proportion of F_sum attributable to diving, last X yrs
  number F_rA_prop;       //proportion of F_sum attributable to headboat, 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);
  number F_end_apex;

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

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

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

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

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

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

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

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

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

  init_bounded_number log_avg_F_cL(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cL(styr_cL_L,endyr_cL_L,-10.0,10.0,2);
  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_cH(-10,0.0,1);
  init_bounded_dev_vector log_F_dev_cH(styr_cH_L,endyr_cH_L,-10.0,10.0,2);
  matrix F_cH(styr,endyr,1,nages);
  vector F_cH_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cH;
  number log_F_dev_end_cH;

  init_bounded_number log_avg_F_rA(-15.0,0.0,1);
  init_bounded_dev_vector log_F_dev_rA(styr_rA_L,endyr_rA_L,-10.0,10.0,2);
  matrix F_rA(styr,endyr,1,nages);
  vector F_rA_out(styr,endyr);       //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_rA;
  number log_F_dev_end_rA;

  init_bounded_number F_init(0.01,0.9,-4);
  //number F_init_ratio;   //scales initial F, which is read in as a fixed value
  vector sel_initial(1,nages);       //initial selectivity (a combination of for-hire and commercial selectivities)

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

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

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


////-------Objective function components---------------------------------------------------------------------
  number w_L;
  number w_lc_cL;
  number w_lc_cH;
  number w_lc_rA;
  //number w_lc_mm;
  number w_ac_cL;
  number w_ac_cH;
  number w_ac_mm;
  number w_I_mm;
```

```
  number w_I_cL;
  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_cL_cpue;
  number f_mm_cpue;

  number f_cL_L;
  number f_cH_L;
  number f_rA_L;

  number f_cL_lenc;
  number f_cH_lenc;
  number f_rA_lenc;
  //number f_mm_lenc;

  number f_cL_agec;
  number f_cH_agec;
  number f_mm_agec;

  number f_cL_RW_cpue; //random walk component of indices

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

  //init_number xdum;
  objective_function_value fval;
  number fval_data;

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

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


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

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

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

// Set values of fixed parameters or set initial guess of estimated parameters
  Linf=set_Linf;
  K=set_K;
  t0=set_t0;
  Linf_f=set_Linf_f;
  K_f=set_K_f;
  t0_f=set_t0_f;
  len_cv_val=set_len_cv;

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

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

  log_q_mm=set_logq_mm;
  log_q_cL=set_logq_cL;
  q_rate=set_q_rate;
  q_rate_fcn_cL=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;
  q_RW_log_dev_cL.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);}
      }
  } //end q_rate conditional

  w_L=set_w_L;
  w_lc_cL=set_w_lc_cL;
```

```
  w_lc_cH=set_w_lc_cH;
  w_lc_rA=set_w_lc_rA;
  //w_lc_mm=set_w_lc_mm;
  w_ac_cL=set_w_ac_cL;
  w_ac_cH=set_w_ac_cH;
  w_ac_mm=set_w_ac_mm;
  w_I_mm=set_w_I_mm;
  w_I_cL=set_w_I_cL;
  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_cH=set_log_avg_F_cH;
  log_avg_F_rA=set_log_avg_F_rA;
  F_init=set_F_init;

  log_R0=set_log_R0;

  selpar_L50_cL=set_selpar_L50_cL;
  selpar_slope_cL=set_selpar_slope_cL;
  //selpar_L502_cL=set_selpar_L502_cL;
  //selpar_slope2_cL=set_selpar_slope2_cL;
  //selpar_min_cL=set_selpar_min_cL;
  //selpar_afull_cL=set_selpar_afull_cL;
  //selpar_peak_cL=set_selpar_peak_cL;
  //selpar_top_cL=set_selpar_top_cL;
  //selpar_ascwid_cL=set_selpar_ascwid_cL;
  //selpar_decwid_cL=set_selpar_decwid_cL;
  //selpar_init_cL=set_selpar_init_cL;
  //selpar_final_cL=set_selpar_final_cL;
  //selpar_L50_cL2=set_selpar_L50_cL2;
  //selpar_slope_cL2=set_selpar_slope_cL2;

  selpar_L50_cH=set_selpar_L50_cH;
  selpar_slope_cH=set_selpar_slope_cH;
  //selpar_L502_cH=set_selpar_L502_cH;
  //selpar_slope2_cH=set_selpar_slope2_cH;
  //selpar_min_cH=set_selpar_min_cH;
  //selpar_afull_cH=set_selpar_afull_cH;
  //selpar_peak_cH=set_selpar_peak_cH;
  //selpar_top_cH=set_selpar_top_cH;
  //selpar_ascwid_cH=set_selpar_ascwid_cH;
  //selpar_decwid_cH=set_selpar_decwid_cH;
  //selpar_init_cH=set_selpar_init_cH;
  //selpar_final_cH=set_selpar_final_cH;

  selpar_L50_rA=set_selpar_L50_rA;
  selpar_slope_rA=set_selpar_slope_rA;
  //selpar_L502_rA=set_selpar_L502_rA;
  //selpar_slope2_rA=set_selpar_slope2_rA;
  //selpar_min_rA=set_selpar_min_rA;
  //selpar_afull_rA=set_selpar_afull_rA;
  //selpar_peak_rA=set_selpar_peak_rA;
  //selpar_top_rA=set_selpar_top_rA;
  //selpar_ascwid_rA=set_selpar_ascwid_rA;
  //selpar_decwid_rA=set_selpar_decwid_rA;
  //selpar_init_rA=set_selpar_init_rA;
  //selpar_final_rA=set_selpar_final_rA;

  selpar_L50_mm=set_selpar_L50_mm;
  selpar_slope_mm=set_selpar_slope_mm;
  //selpar_L502_mm=set_selpar_L502_mm;
  //selpar_slope2_mm=set_selpar_slope2_mm;
  //selpar_min_mm=set_selpar_min_mm;
  //selpar_afull_mm=set_selpar_afull_mm;
  //selpar_peak_mm=set_selpar_peak_mm;
  //selpar_top_mm=set_selpar_top_mm;
  //selpar_ascwid_mm=set_selpar_ascwid_mm;
  //selpar_decwid_mm=set_selpar_decwid_mm;
  //selpar_init_mm=set_selpar_init_mm;
  //selpar_final_mm=set_selpar_final_mm;

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

 SSB_msy_out=0.0;

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

 maturity_f=maturity_f_obs;
 prop_f=prop_f_obs;

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

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

      nsamp_cL_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nsamp_cH_lenc_allyr=missing;
      nsamp_rA_lenc_allyr=missing;
      //nsamp_mm_lenc_allyr=missing;
```

```
      nsamp_cL_agec_allyr=missing;
      nsamp_cH_agec_allyr=missing;
      nsamp_mm_agec_allyr=missing;
      pred_mm_cpue_allyr=missing;
      obs_mm_cpue_allyr=missing;
      mm_cpue_cv_allyr=missing;

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

      for (iyear=1; iyear<=nyr_cH_lenc; iyear++)
         {if (nsamp_cH_lenc(iyear)>=minSS_cH_lenc)
           {nsamp_cH_lenc_allyr(yrs_cH_lenc(iyear))=nsamp_cH_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_rA_lenc; iyear++)
         {if (nsamp_rA_lenc(iyear)>=minSS_rA_lenc)
           {nsamp_rA_lenc_allyr(yrs_rA_lenc(iyear))=nsamp_rA_lenc(iyear);}}

      //for (iyear=1; iyear<=nyr_mm_lenc; iyear++)
      //   {if (nsamp_mm_lenc(iyear)>=minSS_mm_lenc)
      //     {nsamp_mm_lenc_allyr(yrs_mm_lenc(iyear))=nsamp_mm_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cL_agec; iyear++)
         {if (nsamp_cL_agec(iyear)>=minSS_cL_agec)
           {nsamp_cL_agec_allyr(yrs_cL_agec(iyear))=nsamp_cL_agec(iyear);}}

      for (iyear=1; iyear<=nyr_cH_agec; iyear++)
         {if (nsamp_cH_agec(iyear)>=minSS_cH_agec)
           {nsamp_cH_agec_allyr(yrs_cH_agec(iyear))=nsamp_cH_agec(iyear);}}

      for (iyear=1; iyear<=nyr_mm_agec; iyear++)
         {if (nsamp_mm_agec(iyear)>=minSS_mm_agec)
           {nsamp_mm_agec_allyr(yrs_mm_agec(iyear))=nsamp_mm_agec(iyear);}}

      for (iyear=1; iyear<=nyr_mm_cpue; iyear++)
         {
           obs_mm_cpue_allyr(yrs_mm_cpue(iyear))=obs_mm_cpue(iyear);
           mm_cpue_cv_allyr(yrs_mm_cpue(iyear))=mm_cpue_cv(iyear);
         }

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


//fill in F's, Catch matrices, and log rec dev with zero's
  F_cL.initialize();
  L_cL_num.initialize();
  F_cH.initialize();
  L_cH_num.initialize();
  F_rA.initialize();
  L_rA_num.initialize();

  F_cL_out.initialize();
  F_cH_out.initialize();

  sel_cL.initialize();
  sel_cH.initialize();
  sel_rA.initialize();
  sel_mm.initialize();

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


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


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

 R0=mfexp(log_R0);

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

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

 get_length_weight_at_age();
 //cout << "got length, weight, fecundity transitions" <<endl;
 get_reprod();
 //cout << "got repro stuff" << endl;
 get_length_at_age_dist();
 //cout<< "got predicted length at age distribution"<<endl;
```

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


FUNCTION get_length_weight_at_age
  //compute mean length (mm) and weight (whole) at age
    meanlen_TL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));    //total length in mm
    wgt_mt=wgtpar_a*pow(meanlen_TL,wgtpar_b);             //wgt in mt
    wgt_g=wgt_mt/g2mt;                                    //wgt in grams
    wgt_kg=g2kg*wgt_g;                                   //wgt in kilograms
    wgt_klb=mt2klb*wgt_mt;                               //1000 lb of whole wgt
    wgt_lb=mt2lb*wgt_mt;                                 //1000 lb of whole wgt
    meanlen_TL_f=Linf_f*(1.0-mfexp(-K_f*(agebins-t0_f+0.5)));    //total length in mm
    wgt_mt_f=wgtpar_a*pow(meanlen_TL_f,wgtpar_b);             //wgt in mt
    wgt_g_f=wgt_mt_f/g2mt;                                    //wgt in grams
    wgt_kg_f=g2kg*wgt_g_f;                                   //wgt in kilograms
    wgt_klb_f=mt2klb*wgt_mt_f;                               //1000 lb of whole wgt
    wgt_lb_f=mt2lb*wgt_mt_f;                                 //1000 lb of whole wgt
    //gonad_wgt_mt=g2mt*(gwgtpar_a+gwgtpar_b*wgt_g_f);    //gonad wgt in mt
    gonad_wgt_mt=g2mt*mfexp(gwgtpar_a+gwgtpar_b*log(wgt_g_f));    //gonad wgt in mt
    for (iage=1;iage<=nages;iage++)
    {
      if(gonad_wgt_mt(iage)<0){gonad_wgt_mt(iage)=0;}
    }

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

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

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

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

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

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

  lenprob_cL=lenprob;
  lenprob_cH=lenprob;
  lenprob_rA=lenprob;
  //lenprob_mm=lenprob;

  lenprob_cL_all=lenprob_all;
  lenprob_cH_all=lenprob_all;
  lenprob_rA_all=lenprob_all;
  //lenprob_mm_all=lenprob_all;


FUNCTION get_weight_at_age_landings

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    len_cL_mm(iyear)=meanlen_TL;
    gutwgt_cL_klb(iyear)=wgt_klb/gut2whole;
    len_cH_mm(iyear)=meanlen_TL;
    gutwgt_cH_klb(iyear)=wgt_klb/gut2whole;
    len_rA_mm(iyear)=meanlen_TL;
    wgt_rA_klb(iyear)=wgt_klb;
    len_mm_mm(iyear)=meanlen_TL;
    wgt_mm_klb(iyear)=wgt_klb;
  }

FUNCTION get_spr_F0
```

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

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


FUNCTION get_selectivity

  for (iyear=styr; iyear<=endyr; iyear++)
  {
     //sel_cL(iyear)=logistic_joint(agebins, selpar_L50_cL, selpar_slope_cL, selpar_L502_cL, selpar_slope2_cL, selpar_min_cL, selpar_afull_cL);
     //sel_cH(iyear)=logistic_joint(agebins, selpar_L50_cH, selpar_slope_cH, selpar_L502_cH, selpar_slope2_cH, selpar_min_cH, selpar_afull_cH);
     //sel_rA(iyear)=logistic_joint(agebins, selpar_L50_rA, selpar_slope_rA, selpar_L502_rA, selpar_slope2_rA, selpar_min_rA, selpar_afull_rA);
     //sel_mm(iyear)=logistic_joint(agebins, selpar_L50_mm, selpar_slope_mm, selpar_L502_mm, selpar_slope2_mm, selpar_min_mm, selpar_afull_mm);

     sel_cL(iyear)=logistic(agebins, selpar_L50_cL, selpar_slope_cL);
     //if(iyear>endyr_period1){sel_cL(iyear)=logistic(agebins, selpar_L50_cL2, selpar_slope_cL2);}
     sel_cH(iyear)=logistic(agebins, selpar_L50_cH, selpar_slope_cH);
     //sel_rA(iyear)=logistic(agebins, selpar_L50_rA, selpar_slope_rA);
     sel_mm(iyear)=logistic(agebins, selpar_L50_mm, selpar_slope_mm);

     //sel_cL(iyear)=gaussian_double(agebins, selpar_peak_cL, selpar_top_cL, selpar_ascwid_cL, selpar_decwid_cL, selpar_init_cL, selpar_final_cL);
     //sel_cH(iyear)=gaussian_double(agebins, selpar_peak_cH, selpar_top_cH, selpar_ascwid_cH, selpar_decwid_cH, selpar_init_cH, selpar_final_cH);
     //sel_rA(iyear)=gaussian_double(agebins, selpar_peak_rA, selpar_top_rA, selpar_ascwid_rA, selpar_decwid_rA, selpar_init_rA, selpar_final_rA);
     //sel_mm(iyear)=gaussian_double(agebins, selpar_peak_mm, selpar_top_mm, selpar_ascwid_mm, selpar_decwid_mm, selpar_init_mm, selpar_final_mm);
  }
  sel_initial=sel_cL(styr);
  sel_rA=sel_cH;

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  //initialization F is avg from first 2 yrs of observed landings (3 yr average typically preferable, but not so here)
  log_F_dev_init_cL=sum(log_F_dev_cL(styr_cL_L,(styr_cL_L+1)))/2.0;
  log_F_dev_init_cH=sum(log_F_dev_cH(styr_cH_L,(styr_cH_L+1)))/2.0;
  log_F_dev_init_rA=sum(log_F_dev_rA(styr_rA_L,(styr_rA_L+1)))/2.0;

  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_cH_L & iyear<=endyr_cH_L)
    {  F_cH_out(iyear)=mfexp(log_avg_F_cH+log_F_dev_cH(iyear)); //}
    // if (iyear<styr_cH_L) {F_cH_out(iyear)=0.0;}
       F_cH(iyear)=sel_cH(iyear)*F_cH_out(iyear);
       Fsum(iyear)+=F_cH_out(iyear);
    }

    //-------------
    if(iyear>=styr_rA_L & iyear<=endyr_rA_L)
    {  F_rA_out(iyear)=mfexp(log_avg_F_rA+log_F_dev_rA(iyear));//}
    // if (iyear<styr_rA_L){F_rA_out(iyear)=mfexp(log_avg_F_rA+log_F_init_rA);}
       F_rA(iyear)=sel_rA(iyear)*F_rA_out(iyear);
       Fsum(iyear)+=F_rA_out(iyear);
    }

    //Total F at age
    F(iyear)=F_cL(iyear); //first in additive series (NO +=)
    F(iyear)+=F_cH(iyear);
    F(iyear)+=F_rA(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);
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_dev)-
              sum(log_rec_dev(styr_rec_dev,endyr_rec_dev))
              /(nyrs_rec-(endyr_rec_dev-styr_rec_dev)))/(nyrs_rec-(endyr_rec_dev-styr_rec_dev)-1.0);

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


FUNCTION get_numbers_at_age
//Initialization
  S0=spr_F0*R0;
  R_virgin=(R0/((5.0*steep-1.0)*spr_F0))*
                (BiasCor*4.0*steep*spr_F0-spr_F0*(1.0-steep));
```

```
  B0=bpr_F0*R_virgin;
  B0_q_DD=R_virgin*sum(elem_prod(N_bpr_F0(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));


  F_initial=sel_initial*F_init;

//  F_initial=F_init_ratio*set_F_init;

  Z_initial=M+F_initial;


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

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

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

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



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

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

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

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

  SSB(styr)=sum(elem_prod(N_spawn(styr),reprod));
  MatFemB(styr)=sum(elem_prod(N_spawn(styr),reprod2));
  B_q_DD(styr)=sum(elem_prod(N(styr)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));

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

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

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

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


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

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

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

FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_cL_num(iyear,iage)=N(iyear,iage)*F_cL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cH_num(iyear,iage)=N(iyear,iage)*F_cH(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_rA_num(iyear,iage)=N(iyear,iage)*F_rA(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }

    pred_cL_L_knum(iyear)=sum(L_cL_num(iyear))/1000.0;
    pred_cH_L_knum(iyear)=sum(L_cH_num(iyear))/1000.0;
    pred_rA_L_knum(iyear)=sum(L_rA_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),gutwgt_cL_klb(iyear));     //in 1000 lb
    L_cH_klb(iyear)=elem_prod(L_cH_num(iyear),gutwgt_cH_klb(iyear));     //in 1000 lb
    L_rA_klb(iyear)=elem_prod(L_rA_num(iyear),wgt_rA_klb(iyear));     //in 1000 lb

    pred_cL_L_klb(iyear)=sum(L_cL_klb(iyear));
    pred_cH_L_klb(iyear)=sum(L_cH_klb(iyear));
    pred_rA_L_klb(iyear)=sum(L_rA_klb(iyear));
  }


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);}
      }
  } //end q_rate conditional

 //Get density dependence scalar (=1.0 if density independent model is used)
  if (q_DD_beta>0.0)
  {
    B_q_DD+=dzero;
    for (iyear=styr;iyear<=endyr;iyear++)
        {q_DD_fcn(iyear)=pow(B0_q_DD,q_DD_beta)*pow(B_q_DD(iyear),-q_DD_beta);}
          //{q_DD_fcn(iyear)=1.0+4.0/(1.0+mfexp(0.75*(B_q_DD(iyear)-0.1*B0_q_DD))); }
  }


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

 //Commercial longline 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)),gutwgt_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));}
  }

  q_mm=mfexp(log_q_mm);
  for (iyear=1; iyear<=nyr_mm_cpue; iyear++)
  {
    if(iyear<nyr_mm_cpue)
    {
      N_mm(iyear)=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)-2),sel_mm(yrs_mm_cpue(iyear)-2)),wgt_mm_klb(yrs_mm_cpue(iyear)-2));
      N_mm(iyear)+=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)-1),sel_mm(yrs_mm_cpue(iyear)-1)),wgt_mm_klb(yrs_mm_cpue(iyear)-1));
      N_mm(iyear)+=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)),sel_mm(yrs_mm_cpue(iyear))),wgt_mm_klb(yrs_mm_cpue(iyear)));
      N_mm(iyear)+=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)+1),sel_mm(yrs_mm_cpue(iyear)+1)),wgt_mm_klb(yrs_mm_cpue(iyear)+1));
      pred_mm_cpue(iyear)=q_mm*sum(N_mm(iyear))/4;
    }
    if(iyear==nyr_mm_cpue)
    {
      N_mm(iyear)=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)-1),sel_mm(yrs_mm_cpue(iyear)-1)),wgt_mm_klb(yrs_mm_cpue(iyear)-1));
      N_mm(iyear)+=elem_prod(elem_prod(N_mdyr(yrs_mm_cpue(iyear)),sel_mm(yrs_mm_cpue(iyear))),wgt_mm_klb(yrs_mm_cpue(iyear)));
      pred_mm_cpue(iyear)=q_mm*sum(N_mm(iyear))/2;
    }
  }

  for (iyear=1; iyear<=nyr_mm_cpue; iyear++)
  {
     pred_mm_cpue_allyr(yrs_mm_cpue(iyear))=pred_mm_cpue(iyear);  // for graphing purposes only
  }

FUNCTION get_length_comps

 //Commercial longline
  for (iyear=1;iyear<=nyr_cL_lenc;iyear++)
  {
        pred_cL_lenc(iyear)=(L_cL_num(yrs_cL_lenc(iyear))*lenprob_cL)
```

```
                        /sum(L_cL_num(yrs_cL_lenc(iyear)));
 }

 //Commercial handline
 for (iyear=1;iyear<=nyr_cH_lenc;iyear++)
 {
       pred_cH_lenc(iyear)=(L_cH_num(yrs_cH_lenc(iyear))*lenprob_cH)
                      /sum(L_cH_num(yrs_cH_lenc(iyear)));
 }

//All recreational
 for (iyear=1;iyear<=nyr_rA_lenc;iyear++)
 {
       pred_rA_lenc(iyear)=(L_rA_num(yrs_rA_lenc(iyear))*lenprob_rA)
                        /sum(L_rA_num(yrs_rA_lenc(iyear)));
 }

//MARMAP
// for (iyear=1;iyear<=nyr_mm_lenc;iyear++)
//{
//  if(iyear<nyr_mm_lenc)
//  {
//    pred_mm_lenc(iyear)=elem_prod(N(yrs_mm_lenc(iyear)-2),sel_mm(yrs_mm_lenc(iyear)-2))*lenprob_mm;
//    pred_mm_lenc(iyear)+=elem_prod(N(yrs_mm_lenc(iyear)-1),sel_mm(yrs_mm_lenc(iyear)-1))*lenprob_mm;
//    pred_mm_lenc(iyear)+=elem_prod(N(yrs_mm_lenc(iyear)),sel_mm(yrs_mm_lenc(iyear)))*lenprob_mm;
//    pred_mm_lenc(iyear)+=elem_prod(N(yrs_mm_lenc(iyear)+1),sel_mm(yrs_mm_lenc(iyear)+1))*lenprob_mm;
//    pred_mm_lenc(iyear)=pred_mm_lenc(iyear)/sum(pred_mm_lenc(iyear));
//  }
//  if(iyear==nyr_mm_lenc)
//  {
//    pred_mm_lenc(iyear)+=elem_prod(N(yrs_mm_lenc(iyear)-1),sel_mm(yrs_mm_lenc(iyear)-1))*lenprob_mm;
//    pred_mm_lenc(iyear)+=elem_prod(N(yrs_mm_lenc(iyear)),sel_mm(yrs_mm_lenc(iyear)))*lenprob_mm;
//    pred_mm_lenc(iyear)=pred_mm_lenc(iyear)/sum(pred_mm_lenc(iyear));
//  }
//}

FUNCTION get_age_comps


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

 //Commercial handline
 for (iyear=1;iyear<=nyr_cH_agec;iyear++)
 {
   ErrorFree_cH_agec(iyear)=L_cH_num(yrs_cH_agec(iyear))/sum(L_cH_num(yrs_cH_agec(iyear)));
   pred_cH_agec(iyear)=age_error*ErrorFree_cH_agec(iyear);
 }

 //MARMAP longline
 for (iyear=1;iyear<=nyr_mm_agec;iyear++)
 {
   if(iyear<nyr_mm_agec)
   {
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)-2),sel_mm(yrs_mm_agec(iyear)-2));
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)-1),sel_mm(yrs_mm_agec(iyear)-1));
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)),sel_mm(yrs_mm_agec(iyear)));
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)+1),sel_mm(yrs_mm_agec(iyear)+1));
     pred_mm_agec(iyear)=age_error*(ErrorFree_mm_agec(iyear)/sum(ErrorFree_mm_agec(iyear)));
   }
   if(iyear==nyr_mm_agec)
   {
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)-1),sel_mm(yrs_mm_agec(iyear)-1));
     ErrorFree_mm_agec(iyear)=elem_prod(N(yrs_mm_agec(iyear)),sel_mm(yrs_mm_agec(iyear)));
     pred_mm_agec(iyear)=age_error*(ErrorFree_mm_agec(iyear)/sum(ErrorFree_mm_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_cH+
        sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_rA+
        sum(log_F_dev_rA((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_cH_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cH+
        sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_rA_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_rA+
        sum(log_F_dev_rA((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_cH=sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_rA=sum(log_F_dev_rA((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_cH(endyr)*mfexp(log_avg_F_cH+log_F_dev_end_cH)+
          sel_rA(endyr)*mfexp(log_avg_F_rA+log_F_dev_end_rA);

  F_end=F_end_L;
  F_end_apex=max(F_end);

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

```
  wgt_wgted_L_denom=F_cL_prop+F_cH_prop+F_rA_prop;
  wgt_wgted_L_klb=F_cL_prop/wgt_wgted_L_denom*gutwgt_cL_klb(endyr)+
             F_cH_prop/wgt_wgted_L_denom*gutwgt_cH_klb(endyr)+
             F_rA_prop/wgt_wgted_L_denom*wgt_rA_klb(endyr);

FUNCTION get_msy

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

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

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

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


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

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

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

  msy_klb_out=max(L_eq_klb);

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

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

  sigma_rec_dev=sqrt(var_rec_dev); //pow(var_rec_dev,0.5);  //sample SD of predicted residuals (may not equal rec_sigma)
  len_cv=elem_div(len_sd,meanlen_TL);

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

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_klb_yr(iyear)= pred_cL_L_klb(iyear)+pred_cH_L_klb(iyear)+
                               pred_rA_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_cL_L_knum(iyear)+pred_cH_L_knum(iyear)+
                               pred_rA_L_knum(iyear);

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

  L_total_num=(L_rA_num+L_cL_num+L_cH_num); //landings at age in number fish
  L_total_klb=L_rA_klb+L_cL_klb+L_cH_klb;   //landings at age in klb whole weight

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

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

  if(F_msy_out>0)
    {
      FdF_msy=Fapex/F_msy_out;
```

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

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


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

  //static per-recruit stuff

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


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

    F_L_age_spr=F_spr(ff)*sel_wgted_L;

    Z_age_spr=M+F_L_age_spr;

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

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

FUNCTION get_effective_sample_sizes

      neff_cL_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
      neff_cH_lenc_allyr_out=missing;
      neff_rA_lenc_allyr_out=missing;
      //neff_mm_lenc_allyr_out=missing;
      neff_cL_agec_allyr_out=missing;
      neff_cH_agec_allyr_out=missing;
      neff_mm_agec_allyr_out=missing;

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

      for (iyear=1; iyear<=nyr_cH_lenc; iyear++)
         {if (nsamp_cH_lenc(iyear)>=minSS_cH_lenc)
             {neff_cH_lenc_allyr_out(yrs_cH_lenc(iyear))=multinom_eff_N(pred_cH_lenc(iyear),obs_cH_lenc(iyear));}
          else {neff_cH_lenc_allyr_out(yrs_cH_lenc(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_rA_lenc; iyear++)
         {if (nsamp_rA_lenc(iyear)>=minSS_rA_lenc)
             {neff_rA_lenc_allyr_out(yrs_rA_lenc(iyear))=multinom_eff_N(pred_rA_lenc(iyear),obs_rA_lenc(iyear));}
          else {neff_rA_lenc_allyr_out(yrs_rA_lenc(iyear))=-99;}
         }
```

```
      //for (iyear=1; iyear<=nyr_mm_lenc; iyear++)
      //   {if (nsamp_mm_lenc(iyear)>=minSS_mm_lenc)
      //       {neff_mm_lenc_allyr_out(yrs_mm_lenc(iyear))=multinom_eff_N(pred_mm_lenc(iyear),obs_mm_lenc(iyear));}
      //    else {neff_mm_lenc_allyr_out(yrs_mm_lenc(iyear))=-99;}
      //   }

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

      for (iyear=1; iyear<=nyr_cH_agec; iyear++)
         {if (nsamp_cH_agec(iyear)>=minSS_cH_agec)
            {neff_cH_agec_allyr_out(yrs_cH_agec(iyear))=multinom_eff_N(pred_cH_agec(iyear),obs_cH_agec(iyear));}
          else {neff_cH_agec_allyr_out(yrs_cH_agec(iyear))=-99;}
         }

      for (iyear=1; iyear<=nyr_mm_agec; iyear++)
         {if (nsamp_mm_agec(iyear)>=minSS_mm_agec)
            {neff_mm_agec_allyr_out(yrs_mm_agec(iyear))=multinom_eff_N(pred_mm_agec(iyear),obs_mm_agec(iyear));}
          else {neff_mm_agec_allyr_out(yrs_mm_agec(iyear))=-99;}
         }

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


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

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

  f_cL_cpue=0.0;
  f_cL_cpue=lk_lognormal(pred_cL_cpue, obs_cL_cpue, cL_cpue_cv, w_I_cL);
  fval+=f_cL_cpue;
  fval_data+=f_cL_cpue;

  f_mm_cpue=0.0;
  f_mm_cpue=lk_lognormal(pred_mm_cpue, obs_mm_cpue, mm_cpue_cv, w_I_mm);
  fval+=f_mm_cpue;
  fval_data+=f_mm_cpue;

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

  //f_cL_L in 1000 lb ww
  f_cL_L=lk_lognormal(pred_cL_L_klb(styr_cL_L,endyr_cL_L), obs_cL_L(styr_cL_L,endyr_cL_L),
                    cL_L_cv(styr_cL_L,endyr_cL_L), w_L);
  fval+=f_cL_L;
  fval_data+=f_cL_L;

  //f_cH_L in 1000 lb ww
  f_cH_L=lk_lognormal(pred_cH_L_klb(styr_cH_L,endyr_cH_L), obs_cH_L(styr_cH_L,endyr_cH_L),
                    cH_L_cv(styr_cH_L,endyr_cH_L), w_L);
  fval+=f_cH_L;
  fval_data+=f_cH_L;

  //f_rA_L in 1000 fish
  f_rA_L=lk_lognormal(pred_rA_L_knum(styr_rA_L,endyr_rA_L), obs_rA_L(styr_rA_L,endyr_rA_L),
                    rA_L_cv(styr_rA_L,endyr_rA_L), w_L);
  fval+=f_rA_L;
  fval_data+=f_rA_L;

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

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

  //f_cH_lenc
  f_cH_lenc=lk_multinomial(nsamp_cH_lenc, pred_cH_lenc, obs_cH_lenc, nyr_cH_lenc, minSS_cH_lenc, w_lc_cH);
  fval+=f_cH_lenc;
  fval_data+=f_cH_lenc;

  //f_rA_lenc
  f_rA_lenc=lk_multinomial(nsamp_rA_lenc, pred_rA_lenc, obs_rA_lenc, nyr_rA_lenc, minSS_rA_lenc, w_lc_rA);
  fval+=f_rA_lenc;
  fval_data+=f_rA_lenc;

  //f_mm_lenc
  //f_mm_lenc=lk_multinomial(nsamp_mm_lenc, pred_mm_lenc, obs_mm_lenc, nyr_mm_lenc, minSS_mm_lenc, w_lc_mm);
  //fval+=f_mm_lenc;
  //fval_data+=f_mm_lenc;

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

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

  //f_cH_agec
  f_cH_agec=lk_multinomial(nsamp_cH_agec, pred_cH_agec, obs_cH_agec, nyr_cH_agec, minSS_cH_agec, w_ac_cH);
  fval+=f_cH_agec;
  fval_data+=f_cH_agec;

  //f_mm_agec
  f_mm_agec=lk_multinomial(nsamp_mm_agec, pred_mm_agec, obs_mm_agec, nyr_mm_agec, minSS_mm_agec, w_ac_mm);
  fval+=f_mm_agec;
  fval_data+=f_mm_agec;
```

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

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

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


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


  //Random walk components of fishery dependent indices
  f_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;

//---Priors-----------------------------------------------------
//neg_log_prior arguments: estimate, prior, variance, pdf type
//Variance input as a negative value is considered to be CV in arithmetic space (CV=-1 implies loose prior)
//pdf type 1=none, 2=lognormal, 3=normal, 4=beta
  f_priors=0.0;
  f_priors+=neg_log_prior(len_cv_val, set_len_cv, square(set_len_cv_se), 2);
  //f_priors+=neg_log_prior(steep, set_steep, square(set_steep_se), 4);
  f_priors+=neg_log_prior(rec_sigma,set_rec_sigma,square(set_rec_sigma_se),3);
  //f_priors+=neg_log_prior(R_autocorr,set_R_autocorr, 1.0, 1);
  //f_priors+=neg_log_prior(q_DD_beta, set_q_DD_beta, square(set_q_DD_beta_se), 2);
  //f_priors+=neg_log_prior(q_rate, set_q_rate, dzero+square(set_q_rate), 2);
  //f_priors+=neg_log_prior(F_init, set_F_init, -1.0 , 2);
  //f_priors+=neg_log_prior(M_constant, set_M_constant, square(set_M_constant_se), 2);

  //f_priors+=neg_log_prior(selpar_L50_cL, set_selpar_L50_cL, 1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cL,set_selpar_slope_cL, -0.5, 3);
  //f_priors+=neg_log_prior(selpar_L50_cH, set_selpar_L50_cH, 1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_cH,set_selpar_slope_cH, -0.5, 3);
  //f_priors+=neg_log_prior(selpar_L50_mm, set_selpar_L50_mm, 1.0, 3);
  f_priors+=neg_log_prior(selpar_slope_mm,set_selpar_slope_mm, -0.5, 3);

  //f_priors+=neg_log_prior(selpar_L50_rA, set_selpar_L50_rA, 0.5, 2);
  //f_priors+=neg_log_prior(selpar_slope_rA,set_selpar_slope_rA, 0.5, 2);
  //f_priors+=neg_log_prior(selpar_L502_cL, set_selpar_L502_cL, -1.0, 1);
  //f_priors+=neg_log_prior(selpar_slope2_cL, set_selpar_slope2_cL, -1.0, 1);
  //f_priors+=neg_log_prior(selpar_min_cL, set_selpar_min_cL, -1.0, 1);
  //f_priors+=neg_log_prior(selpar_afull_cL, set_selpar_afull_cL, -1.0, 1);

  //f_priors+=neg_log_prior(selpar_peak_cD, set_selpar_peak_cD, -1.0, 2);
  //f_priors+=neg_log_prior(selpar_top_cD, set_selpar_top_cD, -1.0, 2);
  //f_priors+=neg_log_prior(selpar_ascwid_cD, set_selpar_ascwid_cD, -1.0, 2);
  //f_priors+=neg_log_prior(selpar_deswid_cD, set_selpar_deswid_cD, -1.0, 2);
  //f_priors+=neg_log_prior(selpar_init_cD, set_selpar_init_cD, -1.0, 2);
  //f_priors+=neg_log_prior(selpar_final_cD, set_selpar_final_cD, -1.0, 2);

  fval+=f_priors;

  //if(!last_phase())
  //{
    for (iyear=styr; iyear<=endyr; iyear++)
    {
      if(Fapex(iyear)>1.0)
      {
        fval+=10*(mfexp(Fapex(iyear)-1.0)-1.0);
      }
    }
  //}
  //cout << "fval = " << fval << "  fval_data = " << fval_data << endl;
  //cout << endl;
  //cout << "f_cL_U = " << f_cL_cpue << " f_mm_U = " << f_mm_cpue << " f_cL_L = " << f_cL_L << " f_cH_L = " << f_cH_L
  //<< " f_rA_L = " << f_rA_L << endl;
  //cout << "f_cL_lenc = " << f_cL_lenc << " f_cH_lenc = " << f_cH_lenc << " f_rA_lenc = " << f_rA_lenc << " f_mm_lenc = " << f_mm_lenc << endl;
  //cout << "f_cL_agec = " << f_cL_agec << " f_cH_agec = " << f_cH_agec << " f_mm_agec = " << f_mm_agec << endl;
  //cout << "f_rec_dev = " << f_rec_dev << " f_rec_dev_early = " << f_rec_dev_early << " f_rec_dev_end = " << f_rec_dev_end <<  " f_rec_hist_dev = " << f_rec_historic_dev <<  " f_cL_RW = " << f_cL_RW_cpue << end
  //cout << endl;

//--------------------------------------------------------------------------------
//Logistic function: 2 parameters
FUNCTION dvar_vector logistic(const dvar_vector& ages, const dvariable& L50, const dvariable& slope)
```

```
  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1./(1.+mfexp(-1.*slope*(ages-L50))); //logistic;
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

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

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

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

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

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

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

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

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

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

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

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

```
  dvar_vector var(cv.indexmin(),cv.indexmax()); //variance in log space
  var=log(1.0+square(cv/wgt_dat));   // convert cv in arithmetic space to variance in log space
  LkvalTmp=sum(0.5*elem_div(square(log(elem_div((pred+dzero),(obs+dzero)))),var) );
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;

//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
FUNCTION dvariable lk_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const double& ncomp, const double& minSS, const dvariable& wgt_dat)
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp = number of yrs in matrix, minSS=min N threshold, wgt_dat=scaling of N's
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  LkvalTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {if (nsamp(ii)>=minSS)
    {LkvalTmp-=wgt_dat*nsamp(ii)*sum(elem_prod((obs_comp(ii)+dzero),
               log(elem_div((pred_comp(ii)+dzero), (obs_comp(ii)+dzero))));
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;

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


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

  if (last_phase())
  {

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

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

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

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

  } //endl last phase loop
```

Appendix B Data inputs for the Beaufort Assessment Model

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  SEDAR 25 Assessment: Tilefish
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#starting and ending year of model
1962
2010
#Starting year to estimate recruitment deviation from S-R curve
1976
#Ending year to estimate recruitment deviation from S-R curve
2003
#3 phases of constraints on recruitment deviations:
#allows possible heavier constraint (weights defined later) in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1984
2000

# ending year for first selectivity period
2011

#Number of ages (20 classes is 1,...,25+)
25
#vector of agebins, last is a plus group
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0

#number length bins used to match length comps and number used to compute plus group
23
5

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
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

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

#########################################################################################################
########################Commercial Longline fishery landings##########################
#Commercial Longline CPUE Index from Logbook
#Starting and ending years of CPUE index
1993
2010
#Observed CPUE and assumed CVs
0.623027 0.623359 0.765415 0.452756 0.609743 0.712064 0.636855 0.676573 0.635274 0.519392 0.662369 0.589789 1.060471 1.324642 1.907125 1.996987 1.849201 2.354958
0.201383 0.184838 0.198467 0.235836 0.207862 0.235749 0.275943 0.25412 0.216002 0.249123 0.222257 0.258977 0.263964 0.206838 0.220892 0.250192 0.258114 0.194812
#Commercial Longline fishery landings
#Starting and ending years of landings time series, respectively
1962
2010
#Observed landings (1000 lb gutted weight) and assumed CVs
#2.934096251 2.775449848 0.861206897 20.09598346 3.773142723 8.930887646 5.466528869 4.466387646 8.854163555 16.40044897 9.77846487 34.26224603 77.84273176 133.9679944 129.7896354 62.75992475
82.71318654 112.9615222 177.7968836 783.6884656 2774.40433 1630.173717 1108.275682 989.9038362 985.5756345 228.3658546 379.9318601 743.9150486 757.8250937 822.7134594 887.3747445 866.0910336 702.0156581
591.4584251 310.936 328.3384744 334.5741152 473.7713839 666.8575451 389.5741945 368.099 222.2352216 231.8779982 265.752 379.4758715 260.5693969 300.497 300.673 335.747
2.934312236 2.776070875 0.861781587 20.09585081 3.772784483 8.931487332 5.466682792 4.46575229 8.854496658 16.40006562 9.778436487 34.26304891 77.84254715 133.9683953 129.7890303 62.75955574 92.13983137
114.2319402 177.796801 783.6888254 2774.404057 1630.174176 1108.275924 989.9039212 985.5753479 247.3427078 452.7192586 743.9153422 757.8252872 822.7137664 887.3742992 866.0911687 702.0157751 591.4578366
310.9968539 328.3384878 334.5736557 473.7713436 666.8579793 389.5738214 368.1033591 222.2353832 231.8780229 265.7534303 379.4758716 260.5696992 300.5109283 300.7984519 335.8703939
#0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.46 0.42 0.38 0.34 0.3 0.26 0.22 0.18 0.14 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 0.1
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Number and vector of years of length compositions for longline fishery
5
1988 1989 1990 1991 1994
#sample size of commercial length comp data by year    (first row observed N, second row effective N: effective may be set to observed)
#1057 328 738 5291 9943
#number of trips
8 5 7 40 59
#commercial length composition samples (year,lengthbin 3 cm)
0.0020 0.0048 0.0116 0.0373 0.0641 0.0710 0.1224 0.1465 0.1042 0.0866 0.0607 0.0569 0.0491 0.0484 0.0369 0.0252 0.0119 0.0141 0.0150 0.0181 0.0066 0.0052 0.0014
0.0074 0.0037 0.0324 0.0414 0.0635 0.0561 0.1489 0.1208 0.1062 0.0799 0.0541 0.0494 0.0366 0.0332 0.0410 0.0154 0.0095 0.0210 0.0191 0.0369 0.0185 0.0022 0.0030
0.0000 0.0017 0.0042 0.0043 0.0287 0.0499 0.0400 0.0886 0.1237 0.1197 0.0801 0.0797 0.0937 0.0650 0.0592 0.0327 0.0358 0.0146 0.0191 0.0141 0.0245 0.0138 0.0069
0.0064 0.0107 0.0353 0.1034 0.1116 0.0836 0.0475 0.0725 0.0793 0.0959 0.0744 0.0595 0.0511 0.0450 0.0361 0.0245 0.0209 0.0140 0.0111 0.0064 0.0031 0.0026 0.0051
0.0011 0.0013 0.0068 0.0600 0.1494 0.1761 0.1130 0.1264 0.0971 0.0628 0.0354 0.0477 0.0394 0.0244 0.0165 0.0087 0.0109 0.0078 0.0062 0.0032 0.0031 0.0022 0.0007
###Number and vector of years of age compositions for longline fishery
19
1987 1992 1993 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
###sample sizes of age comps by year  (first row observed N, second row effective N: effective may be set to observed)
# took the minimum of (n ages, n lengths)
#28 124 209 373 736 782 238 190 312 234 32 167 264 368 820 945 554 880 703
#number of trips for lengths
7 100 141 64 30 19 15 26 13 23 19 10 15 16 36 35 20 25 24
#age composition samples (year,age)
0 0 0.199304629 0.131251218 0.397934869 0.271509284 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0.004505096 0.075441839 0.258733253 0.184823416 0.120768897 0.153580676 0.042623326 0.044542537 0.015656724 0.033421213 0.013495086 0.0257309 0.019363832 0 0 0.007313205 0 0 0 0 0 0
0 0 0.002909189 0.144585046 0.293670295 0.269535532 0.137318648 0.079539198 0.009731812 0.007846592 0.014771119 0.008543483 0.003795094 0.007471849 0.008454728 0.008245989 9.6973E-05 0 0 0 0.003484452
0 0 0 0
0 0 0 0.013694172 0.115907413 0.227741679 0.28846315 0.205688647 0.104387409 0.014942224 0.007838471 0.002622232 0.007484783 0.002388498 0.005063651 0.000645635 0.001466792 0.00025402 9.26116E-05
0.000183459 0 0 0 0 0.001135154
0 0.002509373 0.000502196 0.008935661 0.030049839 0.128458821 0.210074757 0.248689767 0.166125655 0.079868423 0.036589629 0.015083545 0.012264716 0.011649553 0.004700725 0.011340633 0.012453709
0.008351015 0.0064295 0.002983729 0 0.000801479 0.000481316 0.000593212 0.001062749
0 0 0.002087832 0.026619014 0.07137598 0.137060644 0.218642459 0.190369347 0.153257924 0.088335684 0.036705766 0.028658922 0.014599335 0.008171252 0.001589715 0.004541628 0.000644793 0.004466112
0.003121341 0.00194164 0.00289818 0.002476064 0.000607519 0.000608003 0.001220847
0 0 0.007034164 0.037539127 0.155546555 0.19644317 0.195959873 0.142654102 0.112813108 0.095149408 0.017246649 0.000972618 0.028392606 0.004646282 0 0 0 0 0.000972618 0.000972618 0.000972618 0.002684485
0 0 0
0 0 0.000144186 0.019054176 0.143659692 0.33824587 0.128262433 0.095380822 0.07523624 0.062095491 0.074396356 0.026766323 0.010235402 0.008526798 0 0 0.000883139 0.004606742 0.002789999 0.009240518
0 0.000475814 0 0 0
0 0 0.00255567 0.004303913 0.086407613 0.170833255 0.322869055 0.126416538 0.073042581 0.038298455 0.043168983 0.052326899 0.039592226 0.014417992 0.004177605 0.0023727 0 0.003213179 0.004406612
0.004521116 0.002827173 0.002281806 0.000618555 0 0.001348072
0 0 0 0.017992571 0.069594403 0.117692757 0.246116569 0.196549318 0.098953482 0.066373767 0.059302698 0.036009654 0.029902136 0.005130768 0.004069164 0.00907171 0.009795788 0.00573228 0.00152358
0.000916411 0.000544944 0.00416156 0.00527596 0.003177267 0.012113213
0 0 0 0.023200603 0.059621724 0.058485322 0.239049473 0.182049349 0.11603677 0 0.021602889 0.068892964 0.095243989 0.047605117 0 0 0.024123226 0.038075093 0 0 0.026013479 0 0 0 0
0 0 0.00024251 0.000589239 0.080024211 0.150698769 0.37060902 0.171923383 0.115759326 0.057711309 0.013598584 0.010979078 0.011079289 0.001581324 0.013167678 0 0.000454956 0 0 0 0 0 0 0.001581324
0 0 0.006470537 0.014111417 0.038708322 0.096885705 0.292525069 0.172360531 0.141511401 0.061281337 0.045219895 0.047000852 0.013346509 0.010313137 0 0.003289436 0.006262075 0.008576498 0.00110961
0.001682471 0 0.010414906 0.019723484 0.003176177 0.006030632
0 0 0 0 0.003580391 0.038713454 0.137712978 0.20705835 0.149623859 0.113731579 0.111038684 0.076272402 0.018728638 0.017939014 0.034414704 0.005488413 0.028080604 0.009695942 0.01113266 0.006400614
0 0.001743635 0.002573169 0.006931782 0.019139129
0 0 8.48463E-05 0.012417657 0.047374351 0.120402882 0.151005774 0.227873109 0.164759268 0.090350504 0.074410244 0.025460316 0.028224235 0.010668441 0.007141892 0.009920116 0.009114076 0 0.00198205
0.007674155 0.002970609 0 0.001110895 0.001487277 0.005567301
0 0 0 0.004652522 0.038608724 0.151614049 0.122859908 0.184429522 0.200882761 0.09386775 0.066700969 0.044227313 0.031923611 0.012826288 0.014062924 0.003439688 0.00462487 0.007232201 0.0036035
0.00303121 0.000352824 0.001917433 0.001606961 0.001358094 0.006176878
0 0 0 0.001293395 0.022413664 0.081042698 0.173190347 0.186163428 0.187135657 0.164093246 0.062655084 0.034184233 0.022946676 0.020543918 0.013219857 0.009567044 0.002893724 0.00412696 0.002177327
0.004084268 0.002475853 0.001421149 0.000554358 0.001597742 0.002219373
0 0 0 0 0.011409 0.067877552 0.140738437 0.158021865 0.133624858 0.142302258 0.11482681 0.075342224 0.045029788 0.025260465 0.020028391 0.008981418 0.009969702 0.017073562 0.009397505 0.001271041
0.004017523 0.005460466 0.004368616 0.000974009 0.004024508
0 0 0 0 0.008886347 0.056395869 0.108736826 0.174465139 0.170479386 0.140747412 0.101350545 0.074420529 0.053166823 0.034056647 0.022594424 0.012925081 0.016303365 0.005671052 0.00249701 0.001993887
0.003645214 0.0016489 0.003880194 0.000866714 0.005268635

#####################################################################################################
#################################Commercial handline fishery landings ########### ########### ########### ###########
#Starting and ending years of landings time series, respectively
1962
2010
#Observed landings (1000 lb gutted weight)  and CVs
#0.468903749 0.443550152 0.137793103 3.208016542 0.601857277 1.426112354 0.872471131 0.712612354 1.413836445 2.617551027 1.56053513 5.468753974 12.42526824 21.57000562 21.92836458 25.73407525
54.66281346 18.07847775 148.6041164 334.4075344 596.73167 263.2592832 202.6863183 146.9931638 133.8843655 23.53314544 50.19913985 92.61095138 86.06090635 82.34654057 81.52725549 171.1079664 105.4283419
82.71757493 34.024 34.13352563 28.89088483 38.10361607 54.20445494 38.54980553 57.579 18.75977843 29.12700177 41.426 26.59412853 49.74760312 33.893 27.416 30.192
0.468377764 0.443119125 0.137558413 3.20771919 0.602215517 1.425652668 0.872597208 0.71282771 1.413363342 2.617794376 1.560843513 5.469101087 12.42530285 21.57057465 21.92849966 25.73398426 91.55425863
55.85739976 148.604559 334.4070646 596.7317731 263.2589939 202.6866999 146.9927328 133.8844933 24.75090155 50.22791198 92.61100046 86.06113413 82.34640818 81.52724593 171.1075173 105.4282552 82.71777397
34.03056606 34.13325223 28.89064423 38.10408498 54.20400072 38.54978857 57.57915086 18.75992676 29.12672462 41.42624972 26.59383838 49.74739083 33.89502174 27.42757814 30.20300614
#0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.46 0.42 0.38 0.34 0.3 0.26 0.22 0.18 0.14 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 0.1
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Number and vector of years of length compositions (comm handline)
9
1984 1985 1986 1987 1991 1993 1994 1995 1996
###sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
#19 52 79 58 134 54 68 438 13
#number of trips
2 6 2 2 7 3 2 5 2
#commercial handline length comp samples (year,age) (3cm length bins)
0.0000 0.0000 0.0000 0.0000 0.0000 0.0598 0.0000 0.0000 0.0000 0.1340 0.1794 0.4186 0.0598 0.0598 0.0598 0.0000 0.0144 0.0000 0.0000 0.0000 0.0000 0.0000 0.0144
0.0000 0.0000 0.0000 0.0010 0.0000 0.0010 0.0000 0.0002 0.0007 0.0518 0.1294 0.2919 0.1294 0.0777 0.0556 0.0259 0.0800 0.1294 0.0000 0.0259 0.0000 0.0000 0.0000
0.0000 0.0128 0.0256 0.0000 0.0897 0.0897 0.0641 0.0384 0.0897 0.0513 0.0769 0.1031 0.1025 0.0513 0.0769 0.0256 0.0128 0.0384 0.0384 0.0128 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0345 0.0690 0.0690 0.0000 0.0000 0.0345 0.1035 0.2414 0.3104 0.0690 0.0000 0.0000 0.0345 0.0345 0.0000 0.0000 0.0000
0.0000 0.0000 0.0090 0.0269 0.0269 0.0537 0.0269 0.0805 0.0716 0.0805 0.0537 0.0910 0.1081 0.0753 0.0448 0.0455 0.0805 0.0805 0.0358 0.0000 0.0090 0.0000 0.0000
0.0000 0.0186 0.0186 0.0186 0.0186 0.0186 0.0929 0.0372 0.0372 0.1487 0.0929 0.0743 0.0372 0.0000 0.0743 0.0186 0.0106 0.0186 0.0558 0.0186 0.0417 0.0558 0.0929
0.0000 0.0000 0.0000 0.0121 0.0284 0.2195 0.0774 0.2038 0.1264 0.0732 0.1094 0.0526 0.0405 0.0163 0.0121 0.0121 0.0000 0.0163 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0119 0.0460 0.0657 0.0642 0.1465 0.1216 0.1426 0.0753 0.0763 0.0557 0.0542 0.0573 0.0411 0.0152 0.0191 0.0073 0.0000 0.0000 0.0000 0.0000
0.0000 0.0215 0.0000 0.0108 0.0108 0.0108 0.0000 0.0000 0.1541 0.0108 0.1541 0.0000 0.1649 0.0000 0.1541 0.0000 0.0000 0.1541 0.0000 0.0000 0.0000 0.1541 0.0000
#Number and vector of years of age compositions (handline)
9
1997 1998 1999 2000 2001 2002 2005 2006 2010
#sample sizes of age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
#84 43 35 222 46 160 103 59 13
#number of trips
5 2 8 8 7 13 5 2 2
#handline age comp samples (year,age)
0 0 0 0.008294252 0.088041312 0.133418126 0.15861773 0.123117164 0.262029586 0.154855991 0.026141162 0.01184251 0.023316487 0.003871849 0.005163964 0.001289867 0 0 0 0 0 0 0 0 0
0 0 0 0.017511322 0.070045288 0.158061965 0.053001222 0.17511322 0.210595931 0.140557832 0.140090576 0.017511322 0.017511322 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0.016666667 0 0.335720641 0.218801898 0.077239027 0.103173191 0.218460854 0.022494069 0 0 0 0 0.007443654 0 0 0 0 0 0 0 0 0
0 0 0 0 0.040702291 0.135353467 0.365721391 0.268024159 0.054068731 0.037918396 0.046019155 0.024917732 0.01394569 0.006933521 0 0.006047169 0 0 0.000348299 0 0 0 0 0 0
0 0 0 0 0.001612842 0.001297908 0.108174912 0.400500076 0.214383875 0.11324248 0.003168423 0 0.103679951 0.053939533 0 0 0 0 0 0 0 0 0 0 0
0 0 0.001232395 0.02983957 0.129198725 0.159799572 0.090822866 0.1032997 0.125835377 0.111065351 0.041227528 0.047570465 0.070371341 0.030447968 0.000324479 0.008626768 0.020220646 0 0 0.020220646
0 0 0.009896604 0 0
0 0 0 0.003723747 0.046042214 0.09340108 0.192242774 0.221971846 0.076969033 0.050325683 0.094735133 0.065789092 0.020350102 0.063808314 0.035369794 0.006783367 0.013566735 0 0 0 0 0.014921088
0 0 0
0 0 0 0.005844905 0.108141356 0.317801964 0.268205108 0.180683027 0.064027319 0.029103327 0.002910333 0.005820665 0.008730998 0.005820665 0 0 0 0 0 0.002910333 0 0 0 0 0
0 0 0 0 0 0.180836545 0.241714393 0.059076847 0.062682852 0.062682852 0.149491117 0.151296121 0.031341426 0.029536422 0 0 0.031341426 0 0 0 0 0 0 0 0

########################################Recreational (all sectors) landings########################################
###Starting and ending years for landings time series
1981
2010
#recreational landings vector (1000 lbs whole weight) and CVs
0.412 0.018 3.198733424 0.726496385 47.29314878 0.319163473 0.147554908 3.966982258 0.014 0.3490712 0.390334413 7.273006193 0.02 12.77761655 0.02 3.520248744 29.58323551 1.238094364 8.227015694
14.31350456 35.17903282 17.74164593 45.41947119 7.757966423 28.5071168 51.0761925 9.774837106 0.02 54.51362505 27.74662861
#0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
#Number and vector of years of length compositions for recreational
14
1984 1989 1990 1996 1997 1998 2001 2002 2003 2004 2005 2006 2009 2010
#sample sizes of length comp data by year   (first row observed N, second row effective N: effective may be set to observed)
#10 17 14 53 126 48 20 28 64 28 130 17 12 11
#number of trips
2 10 6 6 9 5 7 6 7 4 9 4 5 5
#recreational length comp samples (year,lengthbin)
```

```
0.1000 0.0000 0.0000 0.0000 0.1000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.1000 0.2000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.1000 0.2000 0.2000
0.2941 0.2353 0.1176 0.1176 0.0000 0.0588 0.0000 0.0000 0.1176 0.0000 0.0000 0.0588 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.8571 0.0714 0.0000 0.0000 0.0000 0.0714 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 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0189 0.0377 0.0189 0.0755 0.1132 0.0377 0.1321 0.1132 0.0566 0.1321 0.0755 0.0566 0.0566 0.0000 0.0189 0.0377 0.0000 0.0000 0.0189
0.0000 0.0317 0.0397 0.0556 0.0794 0.0952 0.1032 0.1270 0.0476 0.1190 0.0635 0.0635 0.0476 0.0079 0.0317 0.0000 0.0159 0.0238 0.0317 0.0000 0.0079 0.0000 0.0079
0.0000 0.0208 0.0208 0.0417 0.0833 0.1042 0.0417 0.0417 0.0833 0.0417 0.0417 0.0417 0.1042 0.0417 0.0417 0.0417 0.0625 0.0000 0.0625 0.0208 0.0208 0.0000 0.0417
0.1000 0.0000 0.1000 0.0000 0.0000 0.3000 0.2000 0.1000 0.1000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0500 0.0000 0.0000 0.0000 0.0500 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0357 0.0357 0.1786 0.0357 0.1786 0.1429 0.1429 0.0714 0.1071 0.0000 0.0000 0.0000 0.0714 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0781 0.1875 0.2969 0.0781 0.1406 0.0938 0.1094 0.0156 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.1071 0.3571 0.1071 0.0357 0.0000 0.1429 0.2143 0.0000 0.0000 0.0357 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
0.0000 0.0000 0.0000 0.0615 0.0846 0.3538 0.2154 0.1308 0.0692 0.0462 0.0231 0.0154 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0588 0.2353 0.0588 0.1176 0.0588 0.1176 0.1176 0.0588 0.0000 0.0588 0.0000 0.0000 0.0588 0.0000 0.0000 0.0000 0.0588 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0833 0.0000 0.1667 0.1667 0.0833 0.2500 0.0000 0.0000 0.1667 0.0000 0.0000 0.0000 0.0000 0.0833 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0909 0.0000 0.1818 0.0909 0.0000 0.0909 0.0000 0.1818 0.1818 0.0909 0.0000 0.0909 0.0000 0.0000 0.0000 0.0000 0.0000

########################################MARMAP data#########################################
#Number and vector of years of MARMAP Longline CPUE index
5
1985 1998 2002 2006 2010
#Observed CPUE and assumed CVs
1.03 2.46 1.00 0.89 2.82
1.63 1.59 1.75 1.92 1.51 # computed as reported sd/cpue
#Number and vector of years of age compositions (MARMAP)
5
1985 1998 2002 2006 2010
#sample sizes of age comp data by year   (first row observed N, second row effective N: effective may be set to observed)
#269 344 94 80 332
#number of sets
59 38 23 14 44
#MARMAP age comp samples (year,age)
0 0 0 0.022304833 0.089219331 0.081784387 0.104089219 0.118959108 0.141263941 0.137546468 0.118959108 0.044609665 0.026022305 0.018587361 0.007434944 0.011152416 0 0.022304833 0.007434944 0.007434944
0.011152416 0.003717472 0.007434944 0.003717472 0.011152416
0 0 0 0.002906977 0.00872093 0.049418605 0.113372093 0.171511628 0.206395349 0.183139535 0.113372093 0.055232558 0.063953488 0.020348837 0.011627907 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0.021276596 0.021276596 0.127659574 0.244680851 0.159574468 0.180851064 0.117021277 0.074468085 0.042553191 0 0 0.010638298 0 0 0 0 0 0 0
0 0 0 0 0 0 0.0125 0.0625 0.125 0.175 0.1375 0.225 0.1 0.1 0.0375 0.0125 0 0.0125 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0.036144578 0.111445783 0.153614458 0.225903614 0.147590361 0.138554217 0.063253012 0.054216867 0.015060241 0.018072289 0.009036145 0.015060241 0.006024096 0.003012048 0.003012048
0 0 0 0


#####################Biological input ########################################################################
#VonBert params (Linf, K, t0), units in mm TL for all fish
825.1
0.189
-0.47
#VonBert params (Linf, K, t0), units in mm TL for all females only
806.3
0.167
-0.47
#Standard errors of vonBert param (Linf, K, t0), applied if params are estimated
82.51
0.0189
0.047
#CV of length at age
0.15
#standard error of CV of length at age, applied if CV is estimated
0.03

#length-weight (TL-whole wgt (mt)) coefficients a and b, W=aL^b, (W in mt, TL in mm)
4.04E-12
3.155
#weight-gonad weight (whole wgt-gondad weight) coefficients a and b, GW=a+b*W (units=g)
#-52.20
#0.0541
-9.16802
1.70498
#gutted weight to whole weight conversion
1.05893
#time-invariant vector of % maturity-at-age for females (ages 1-25)
0.1 0.25 0.5 1.00 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

#time-invariant vector of proportion female (ages 1-25)
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
#time of year (as fraction) for spawning: end of May=5/12
0.42
#age-dependent natural mortality at age
0.296792393 0.217430236 0.178339521 0.155367446 0.140459983 0.130157753 0.122726054 0.11719808 0.112991856 0.109735893 0.107181879 0.105157598 0.103539958 0.102238784 0.101186644 0.100332253 0.099636049
0.099067148 0.098601207 0.098218874 0.097904663 0.09764611 0.097433134 0.097257551 0.097112693
#age-independent natural mortality (used only to compute MSST=(1-M)SSBmsy)
0.1083
#SE of age-independent natural mortality
0.03
#Max observed age
40
#Spawner-recruit parameters
#steepness (fixed or initial guess) (0.75 from meta-analysis)
0.84  #mu=alpha/(alpha+beta);  alpha and beta from beta distn
#SE of steepness (from meta-analysis)
0.14  #var=(alpha*beta)/[(alpha+beta)^2*(alpha+beta+1)];  alpha and beta from beta distn
#log_R0 - log virgin recruitment
13.0
# R autocorrelation
0.0
# SD of recruitment in log space
0.4428
# SE of SD recruitment
0.15

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

#commercial longline selectivity starting values
```

```
6.0  # age at 50% selectivity
3.0  # slope of ascending limb
#20.0 # descending age at 50% selection
#0.0  # slope of descending limb
#1.0  # minimum selectivity
#6.0  # first age at full selection
#7.0   #peak=ascending inflection location (as logistic),
#-1.0  #top=width of plateau,
#2.5   #ascwid=ascent width (as log(width))
#4.3   #decwid=descent width (as log(width))
#-7.9  #init
#5.0   #final
#8.0    # age at 50% selection (2nd period)
#1.5    # slope of ascending limb (2nd period)

#commercial handline selectivity starting values
6.0  # age at 50% selectivity
3.0  # slope of ascending limb
#20.0 # descending age at 50% selection
#0.0  # slope of descending limb
#1.0  # minimum selectivity
#6.0  # first age at full selection
#7.0   #peak=ascending inflection location (as logistic),
#-1.0  #top=width of plateau,
#2.5   #ascwid=ascent width (as log(width))
#4.3   #decwid=descent width (as log(width))
#-7.9  #init
#5.0   #final

#recreational selectivity starting values
6.0  # age at 50% selectivity
2.0  # slope of ascending limb
#20.0 # descending age at 50% selection
#0.0  # slope of descending limb
#1.0  # minimum selectivity
#6.0  # first age at full selection
#7.0   #peak=ascending inflection location (as logistic),
#-1.0  #top=width of plateau,
#2.5   #ascwid=ascent width (as log(width))
#4.3   #decwid=descent width (as log(width))
#-7.9  #init
#5.0   #final

#MARMAP selectivity starting values
6.0  # age at 50% selectivity
3.0  # slope of ascending limb
#20.0 # descending age at 50% selection
#0.0  # slope of descending limb
#1.0  # minimum selectivity
#6.0  # first age at full selection
#7.0   #peak=ascending inflection location (as logistic),
#-1.0  #top=width of plateau,
#2.5   #ascwid=ascent width (as log(width))
#4.3   #decwid=descent width (as log(width))
#-7.9  #init
#5.0   #final

###################Likelihood Component Weighting#################################################################################
##Weights in objective fcn
1.0      #landings
0.114  #cL length comps
1.591       #cH length comps
0.0  #rA length comps
#1.0         #mm length comps
0.039  #cL age comps
1.46  #cH age comps
0.073         #mm age comps
8.71  #mm index
3.0  #cL index
1.0     #S-R residuals
0.0 #constraint on early recruitment deviations
0.0     #constraint on ending recruitment deviations
0.0     #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization)
0.0     #weight on tuning F (penalty not applied in final phase of optimization)
#0.0 #penalty on deviation in CV at age
#0.0     #penalty on first difference in CV at age

#log catchabilities (initial guesses)
-8.0      #MARMAP longline CPUE
-8.0      #commercial logbook CPUE
#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
-1
##annual positive rate of increase on all fishery dependent q's due to technology creep
0.0
# DD q switch: Integer value (choose estimation phase, negative value turns it off)
-1
##density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)
0.0
##SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)
0.128
#Age to begin counting D-D q (should be age near full exploitation)
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

##log mean F's (initial guesses)
-5.0 #commercial longline
-5.5 #comm handline
-6.5 #recreational

#Initial F (Input here, could also use mean of first few years)
0.01
```

```
#Tuning F (not applied in last phase of optimization)
0.2
#Year for tuning F
2006

##threshold sample sizes for length comps (set to 99999.0 if sel is fixed)
1.0 #cL
1.0 #cH
1.0 #rA
#1.0 #mm

#threshold sample sizes (greater than or equal to) for age comps
1.0 #cL
1.0 #cH
1.0 #mm

#Ageing error matrix (columns are true age 1-25, rows are ages as read for age comps: columns should sum to one)
1 1.21674E-12 0.000138245 3.98227E-05 3.80014E-06 3.04762E-07 1.65774E-08 6.97239E-10 2.91631E-11 7.67427E-13 2.19448E-14 4.20047E-16 8.38542E-18 1.32313E-19 1.57241E-21 1.87743E-23 2.2826E-25
1.94823E-27 1.65727E-29 9.22312E-32 4.98883E-34 2.67307E-36 1.47043E-38 5.14917E-41 1.98284E-43
0 1 0.092915419 0.008203951 0.000650259 4.96291E-05 2.95815E-06 1.42989E-07 6.66902E-09 2.11376E-10 6.91767E-12 1.58937E-13 3.67438E-15 6.80654E-17 9.66359E-19 1.34641E-20 1.87078E-22 1.87201E-24
1.83586E-26 1.21286E-28 7.68159E-31 4.75828E-33 2.99047E-35 1.23858E-37 5.58598E-40
0 1.21674E-12 0.813892668 0.200612753 0.025603013 0.002606191 0.000205664 1.29284E-05 7.39139E-07 3.00616E-08 1.19008E-09 3.4169E-11 9.4871E-13 2.12493E-14 3.69106E-16 6.13551E-18 9.94585E-20
1.18634E-21 1.36244E-23 1.08181E-25 8.11791E-28 5.87939E-30 4.26736E-32 2.10782E-34 1.1223E-36
0 2.19178E-48 0.092915419 0.582286923 0.231960431 0.044133735 0.00557101 0.000515354 3.97033E-05 2.20754E-06 1.11732E-07 4.17373E-09 1.44336E-10 4.02584E-12 8.76198E-14 1.77659E-15 3.42995E-17
4.95839E-19 6.77368E-21 6.54489E-23 5.88815E-25 5.04262E-27 4.27274E-29 2.53786E-31 1.60812E-33
0 5.8451E-108 0.000138245 0.200612753 0.483564988 0.24100681 0.058795547 0.009057048 0.001033624 8.37034E-05 5.72485E-06 2.89667E-07 1.29391E-08 4.62875E-10 1.29268E-11 3.26876E-13 7.67288E-15
1.3668E-16 2.25614E-18 2.68573E-20 2.93126E-22 3.00208E-24 3.00178E-26 2.16187E-28 1.64335E-30
0 2.3078E-191 2.68073E-09 0.008203951 0.231960431 0.424406659 0.241763503 0.070175785 0.013041743 0.001638768 0.00016008 1.14224E-05 6.83479E-07 3.22971E-08 1.18528E-09 3.82156E-11 1.11341E-12
2.48486E-14 5.03428E-16 7.47534E-18 1.00154E-19 1.2406E-21 1.47971E-23 1.30291E-25 1.19767E-27
0 1.3489E-298 6.77482E-16 3.98227E-05 0.025603013 0.24100681 0.387322602 0.239721872 0.079752598 0.0165665 0.002442862 0.000255917 2.12734E-05 1.3676E-06 6.75446E-08 2.83897E-09 1.04805E-10 2.97941E-12
7.5256E-14 1.41126E-15 2.3487E-17 3.55861E-19 5.118E-21 5.55553E-23 6.2251E-25
0 0 2.23143E-24 2.29446E-08 0.000650259 0.044133735 0.241763503 0.361033738 0.236368913 0.086473511 0.020344459 0.003257803 0.000390157 3.51436E-05 2.3922E-06 1.34011E-07 6.39928E-09 2.35609E-10
7.53662E-12 1.80715E-13 3.78029E-15 7.08553E-17 1.24208E-18 1.67595E-20 2.30757E-22
0 0 9.57879E-35 1.56919E-12 3.80014E-06 0.002606191 0.058795547 0.239721872 0.339525345 0.233063621 0.092465542 0.023563147 0.0042163 0.000548062 5.26556E-05 4.01959E-06 2.53459E-07 1.22881E-08
5.05643E-10 1.56961E-11 4.17603E-13 9.79278E-15 2.11508E-16 3.57703E-18 6.10045E-20
0 0 5.35894E-47 1.27383E-17 5.11012E-09 4.96291E-05 0.00557101 0.070175785 0.236368913 0.324343317 0.22935076 0.096833367 0.026848172 0.00518688 0.000720327 7.66098E-05 6.51194E-06 4.22679E-07
2.2727E-08 9.2469E-10 3.16623E-11 9.39467E-13 2.52713E-14 5.40143E-16 1.15019E-17
0 0 3.90741E-61 1.22741E-23 1.58118E-12 3.04762E-07 0.000205664 0.009057048 0.079752598 0.233063621 0.310460919 0.226099747 0.10073686 0.029790441 0.006124267 0.000927786 0.000108527 9.58891E-06
6.84342E-07 3.69496E-08 1.64764E-09 6.25604E-11 2.11863E-12 5.77059E-14 1.54659E-15
0 0 3.71313E-77 1.40383E-30 1.12577E-16 6.03502E-10 2.95815E-06 0.000515354 0.013041743 0.086473511 0.22935076 0.299956602 0.222716621 0.10383452 0.032360678 0.007139582 0.001173262 0.000143469
1.3805E-05 1.00146E-06 5.88471E-08 2.89174E-09 1.24626E-10 4.36171E-12 1.48314E-13
0 0 4.59867E-95 1.90581E-38 1.84431E-21 3.85382E-13 1.65774E-08 1.29284E-05 0.001033624 0.0165665 0.092465542 0.226099747 0.29013984 0.21963455 0.106272301 0.034910717 0.008227667 0.001415735
0.000186565 1.84105E-05 1.44254E-06 9.27813E-08 5.14383E-09 2.33248E-10 1.01435E-11
0 0 7.4228E-115 3.07105E-47 6.95246E-27 7.93594E-17 3.61951E-11 1.42989E-07 3.97033E-05 0.001638768 0.020344459 0.096833367 0.222716621 0.281938007 0.216901007 0.108469067 0.037426976 0.009213755
0.0016891 0.000229565 2.42701E-05 2.06635E-06 1.48967E-07 8.82479E-09 4.94758E-10
0 0 1.5615E-136 5.87406E-57 6.0306E-33 5.26986E-21 3.07906E-14 6.97239E-10 7.39139E-07 8.37034E-05 0.002442862 0.023563147 0.10073686 0.21963455 0.275132607 0.21414808 0.110438129 0.03954793 0.01024502
0.001941585 0.000280256 3.1944E-05 3.02706E-06 2.3622E-07 1.72107E-08
0 0 4.2811E-160 1.33362E-67 1.20365E-39 1.12848E-25 1.02052E-17 1.49892E-12 6.66902E-09 2.20754E-06 0.00016008 0.003257803 0.026848172 0.10383452 0.216901007 0.268648 0.211387655 0.111954895 0.041629568
0.011138216 0.00222116 0.000342781 4.31597E-05 4.47356E-06 4.26975E-07
0 0 1.5297E-185 3.59394E-79 5.52789E-47 7.79263E-31 1.31784E-21 1.42068E-15 2.91631E-11 3.00616E-08 5.72485E-06 0.000255917 0.0042163 0.029790441 0.106272301 0.21414808 0.262462028 0.209023295
0.113324201 0.043339502 0.012082195 0.002553207 0.000431778 5.99397E-05 7.55451E-06
0 0 7.1238E-213 1.14961E-91 5.84165E-55 1.73527E-36 6.63038E-26 5.93656E-19 6.18078E-14 2.11376E-10 1.11732E-07 1.14224E-05 0.000390157 0.00518688 0.032360678 0.108469067 0.211387655 0.257382228
0.2066689 0.114382946 0.045107907 0.013200708 0.003030875 0.0005682 9.53256E-05
0 0 4.3237E-242 4.3649E-105 1.42046E-63 1.24608E-42 1.29972E-30 1.09369E-22 6.34875E-17 7.67427E-13 1.19008E-09 2.89667E-07 2.12734E-05 0.000548062 0.006124267 0.034910717 0.110438129 0.209023295
0.252498779 0.204761211 0.115584719 0.047375131 0.014928009 0.003810778 0.000857853
0 0 3.4201E-273 1.9672E-119 7.94773E-73 2.88548E-49 9.92654E-36 8.8832E-27 3.1606E-20 1.43866E-15 6.91767E-12 4.17373E-09 6.83479E-07 3.51436E-05 0.000720327 0.007139582 0.037426976 0.111954895
0.2066689 0.248624064 0.203277388 0.118017272 0.051589524 0.018082176 0.005505733
0 0 3.5258E-306 1.0523E-134 1.02323E-82 2.15469E-56 2.9538E-41 3.18102E-31 7.62583E-24 1.39258E-18 2.19448E-14 3.4169E-11 1.29391E-08 1.3676E-06 5.26556E-05 0.000927786 0.008227667 0.03954793
0.113324201 0.204761211 0.245368394 0.20407164 0.125097022 0.060703378 0.02520097
0 0 0 6.682E-151 3.03126E-93 5.18855E-64 3.42452E-47 5.02207E-36 8.91746E-28 6.96014E-22 3.79918E-17 1.58937E-13 1.44336E-10 3.22971E-08 2.3922E-06 7.66098E-05 0.001173262 0.009213755 0.041629568
0.114382946 0.203277388 0.244941111 0.212842445 0.144178375 0.082265712
0 0 0 5.0362E-168 2.0663E-104 4.02904E-72 1.54687E-53 3.49558E-41 5.05396E-32 1.79621E-25 3.5895E-20 4.20047E-16 9.4871E-13 4.62875E-10 6.75446E-08 4.01959E-06 0.000108527 0.001415735 0.01024502
0.043339502 0.115584719 0.20407164 0.254094538 0.242277167 0.191522593
0 0 0 4.5054E-186 3.241E-116 1.00891E-80 2.72236E-60 1.07269E-46 1.38822E-36 2.3935E-29 1.85083E-23 6.30749E-19 3.67438E-15 4.02584E-12 1.18528E-09 1.34011E-07 6.51194E-06 0.000143469 0.0016891
0.011138216 0.045107907 0.118017272 0.212842445 0.288038102 0.317995303
0 0 0 4.7843E-205 1.1697E-128 8.14693E-90 1.86669E-67 1.45128E-52 1.84808E-41 1.64684E-33 5.20814E-27 5.38143E-22 8.38542E-18 2.12493E-14 1.29268E-11 2.83897E-09 2.53459E-07 9.58891E-06 0.000186565
0.001941585 0.012082195 0.047375131 0.125097022 0.242277167 0.376548511

999 #end of data file flag
```