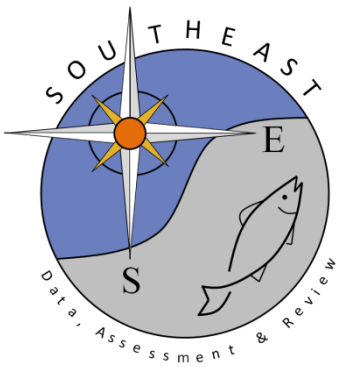
The Beaufort Assessment Model (BAM) with application to Gulf menhaden: mathematical description, implementation details, and computer code

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

SEDAR32A-RW-01

Submitted: 9 August 2013



This information is distributed solely for the purpose of pre-dissemination peer review. It does not represent and should not be construed to represent any agency determination or policy.

Please cite this document as:

Sustainable Fisheries Branch, National Marine Fisheries Service, Southeast Fisheries Science Center – Beaufort Lab. 2013. The Beaufort Assessment Model (BAM) with application to Gulf menhaden: mathematical description, implementation details, and computer code. SEDAR32A-RW01. SEDAR, North Charleston, SC. 37 pp.

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

Sustainable Fisheries Branch

National Marine Fisheries Service

Southeast Fisheries Science Center

NOAA Beaufort Laboratory

101 Pivers Island Road, Beaufort, NC 28516

1 Overview

The primary model in this assessment was the Beaufort assessment model (BAM), which applies a statistical catch-age formulation. The model was implemented with the AD Model Builder software (Fournier et al. 2012), and its structure and equations are detailed herein. In essence, a statistical catch-age model simulates a population forward in time while including fishing processes (Quinn and Deriso 1999; Shertzer et al. 2008). Quantities to be estimated are systematically varied until characteristics of the simulated population match available data on the real population. Statistical catch-age models share many attributes with ADAPT-style tuned and untuned VPAs.

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

2 Model configuration and equations

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

Stock dynamics In the assessment model, new biomass was acquired through growth and recruitment, while abundance of existing cohorts experienced exponential decay from fishing and natural mortality. The population was assumed closed to immigration and emigration. The model included age classes $0 - 4^+$, where the oldest age class 4^+ allowed for the accumulation of fish (i.e., plus group).

Initialization Initial (1977) abundance at age was computed in the model assuming an equilibrium age structure and fishing mortality rate. The equilibrium age structure was computed for ages $1 - 4^+$ based on natural and fishing mortality (F), where F was set equal to the geometric mean fishing mortality from the first three assessment years (1977-1979). In addition, deviations from the equilibrium age structure were also estimated for each age 1 through 4+. The deviations were informed by the age composition data available in the first year of the assessment. Finally, initial age-0 abundance was computed in the model using the Beverton–Holt stock-recruitment curve plus an estimated annual recruitment deviation.

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 M_a to mean weight at age W_a by the power function $\mathrm{M}_a{=}\alpha W_a^{\beta}$, where α is a scale parameter and β is a shape parameter. Lorenzen (1996) provided point estimates of α and β for oceanic fishes, which were used for this assessment. The Lorenzen version of M was scaled to 1.10 at age-2, which is the estimated natural mortality rate based on the tagging study completed by Arhenholz.

Growth Mean size at age of the fishery and the population (fork length, FL) were modeled with the von Bertalanffy equation, and weight at age of the fishery and the population were modeled as a function of FL. Weight at age of the fishery and population were estimated during the assessment process and were treated as an input to the model. For fitting length composition data, a von Bertalanffy growth curve was estimated within the model where the distribution of size at age was assumed normal with a constant coefficient of variation (CV) estimated by the assessment model.

Female maturity Females were modeled to be fully mature at ages-2+ and not mature for ages-0 and -1.

Spawning stock Spawning stock was modeled using total fecundity (mature ova) at the time of peak spawning. For Gulf menhaden, peak spawning was considered to occur January 1.

Recruitment Expected recruitment of age-0 fish was predicted from spawning stock in fecundity using the Beverton–Holt spawner-recruit model. Annual variation in recruitment was assumed to occur with lognormal deviations for the years 1977–2011.

Landings The model included one time series of combined landings from 1977-2011. Reduction, bait, and recreational landings were pooled outside of the model and were entered as one data stream. Landings consisted of mostly commercial reduction fishery landings, which made up about 99% of the total landings. The landings were modeled with the Baranov catch equation (Baranov 1918) and were fitted in units of weight (1,000s metric tons).

Fishing Mortality For the time series of removals, the assessment model estimated an annual full fishing mortality rate (F). Age-specific rates were then computed as the product of full F and selectivity at age.

Selectivities The selectivity curve for the gill net survey CPUE series was estimated using a parametric approach, while the selectivity for the seine CPUE and the landings were not. The parametric approach applies plausible structure on the shape of the curve and achieves greater parsimony than occurs with unique parameters for each age. Selectivity of the gill net index was modeled as flat-topped, using a two parameter logistic function. The selectivity for the recruitment index based on the state seine data was fixed with selectivity of age-0 being 1.0, while the selectivity of all the other ages was 0.0. The selectivity for the fishery was estimated for age-1 but was fixed for all other ages. Specifically, the selectivity was fixed at 0.0 for age-0, 1.0 for age-2, and 0.35 for ages-3 and -4.

Indices of abundance The model was fit to two indices of relative abundance: the Louisiana gill net CPUE index (1988-2011) and the seine recruitment CPUE index (1996-2010). Predicted indices were conditional on selectivities and were computed from abundance at the midpoint of the year for the gill net index and for April 1 for the seine index.

Catchability In the BAM, catchability scales indices of relative abundance to estimated population abundance at large. Several options for time-varying catchability can be implemented in the BAM following recommendations of the 2009 SEDAR procedural workshop on catchability (SEDAR Procedural Guidance 2009). Parameters for each option could be estimated or fixed based on *a priori* considerations. For the base model, the AW assumed time-invariant catchability for both the gill net and seine indices. Because both of these indices are based on consistent, fishery-independent sampling, a constant catchability value was a reasonable assumption.

Biological reference points No benchmarks have been formally adopted for Gulf menhaden. The Menhaden Advisory Committee is discussing goals and objectives for the fishery, after which benchmarks can be discussed. Therefore, for the current assessment, a suite of benchmark options was provided. None of the benchmark options is preferred.

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). However, the value of MSY that resulted was infinite. This characteristic was explored and appears to be infinite because Gulf menhaden are not harvested until after they mature and reproduce. Additional benchmark options were therefore provided and included F_{MED} and a series of benchmarks based on spawners per recruit. In this assessment, spawning stock measures total fecundity in mature ova. These benchmarks are conditional on the estimated selectivity functions.

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 robust multinomial likelihoods.

The model includes the capability for each component of the likelihood to be weighted by user-supplied values (for instance, to give more influence to stronger data sources). For data components, these weights were applied by either adjusting CVs (lognormal components) or adjusting effective sample sizes (multinomial components). In this application to Gulf menhaden, the CV of removals (in arithmetic space) was assumed equal to 0.04, to achieve a close fit to this time series yet allow some imprecision. In practice, the small CVs are a matter of computational

convenience, as they help achieve the desired result of close fits to the landings, while avoiding having to solve the Baranov equation iteratively. Weights on other data components (indices, age and length compositions) were adjusted iteratively, starting from initial weights as follows. The CVs of indices were set equal to the values estimated by jackknifing as reported in the stock assessment report. Effective sample sizes of the annual length compositions were assumed equal to the annual number of sets sampled. Number of annual trips sampled was the effective sample size for the age composition data. These initial weights were then adjusted until standard deviations of normalized residuals (SDNRs) were near 1.0 (SEDAR24-RW03, SEDAR25-RW05, Francis 2011). Computed SDNRs accounted for potential correlations in the composition data (TA1.8 in Table A1 of (Francis 2011)).

The compound objective function included some penalties on the recruitment time series 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, Gulf menhaden 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 Gulf menhaden stock dynamics.

References

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

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

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

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

Fournier, D. A., H. J. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert. 2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software **27**:233–249.

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

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

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

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

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

Pella, J. J., and P. K. Tomlinson. 1969. A generalized stock production model. Bulletin of the Inter-American Tropical Tuna Commission **13**:419–496.

Quinn, T. J., and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York, New York.

SEDAR 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.

Table 2.1. General definitions, input data, population model, and negative log-likelihood components of the statistical catch-age model applied to Gulf menhaden. 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	$y \in \{1977 \dots 2011\}$
Index of ages	a	$a \in \{0, 1 \dots A\}$, where $A = 4^+$
Index of length bins	l	$l \in \{1, 2 \dots 15\}$
Length bins	l'	$l' \in \{85, 95, \dots, 295\text{mm}\}$, with midpoint of 10 mm bin used to match length compositions. Largest 7 length bins ($FL \geq 225$ mm) treated as a plus group, but retained for weight calculations.
Index of fishery	f	f is the combined commercial reduction, bait, and recreational landings
Index of CPUE	u	$u \in \{1, 2\}$ where 1 = Louisiana gill net index, 2 = seine index
Input Data		
Observed length compositions	$p^{\lambda}_{(u=1),l,y}$	Proportional contribution of length bin l in year y to index $u = 1$
Observed age compositions	$p^{\alpha}_{(f),a,y}$	Proportional contribution of age class a in year y to the fishery f
Ageing error matrix	$\mathcal{E}$	Estimated from a comparison between scales and otoliths.
Length comp. sample sizes	$n^{\lambda}_{(u=1),y}$	Effective number of length samples collected in year y from an index u
Age comp. sample sizes	$n^{\alpha}_{(f),y}$	Effective number of age samples collected in year y from the fishery f
Observed landings	$L_{f,y}$	Reported landings in year y from the fishery f. Landings are reported in 1000s of metric tons
CVs of landings	$c^{L}_{f,y}$	Assumed 0.04 in arithmetic space
Observed abundance indices	$U_{u,y}$	$u = 1$, Louisiana gill net index (numbers), $y \in \{1988 \dots 2011\}$ $u = 2$, seine index (numbers), $y \in \{1996 \dots 2010\}$ Annual values estimated from delta-lognormal GLM. Each time series was scaled to its mean.
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1, 2\}$ as above.
Natural mortality rate	M_a	Function of weight at age (w_a): $M_a = \alpha w_a^{\beta}$, with estimates of α and β from Lorenzen (1996). Lorenzen M_a then rescaled such that age-2 mortality was 1.10 as estimated in the tagging study.
Population Model		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages
Proportion females mature at age	m_a	Increasing with age {0, 0, 1 …,1} for ages $0 - 4^+$; 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 0.0 for Gulf menhaden by assuming peak spawning occurs January 1.
Annual fecundity at age	$\mathcal{F}_a$	$\mathcal{F}_a = 0.000051604\ell_a^{3.87751629}$ based on equation provided in Lewis and Roithmayr and was a model input
Mean length at age for the population	l_a	Fork length (January 1); $l_a = L_\infty(1 - \exp[-K(a - t_0)])$ where K, L_∞, and t_0 are parameters estimated by the DW
Mean length at age for the fishery	l_a^f	Fork length (midyear); $l_a = L_\infty(1 - \exp[-K(a - t_0 + 0.5)])$ where K, L_∞, and t_0 are parameters estimated in the assessment model and used for the age-length transition for the length compositions
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.
Individual weight at age of population	w_a	Computed from length at age by $w_a = \theta_1 l_a^{\theta_2}$ where θ_1 and θ_2 are parameters from the DW
Individual weight at age of landings	$w_{(f),a,y}^L$	Computed from length at age by $w_{(f),a,y}^L = \theta_1(\xi_{(f),a,y}^L)^{\theta_2}$. With weight at age of landings being a model input.
Index selectivity	$s_{(u),a}$	$s_{(u),a} = \frac{1}{1+\exp\left[-\widehat{\eta}_{(u)}\left(a-\widehat{\alpha}_{(u)}\right)\right]}$ where $\widehat{\eta}_{(u)}$ and $\widehat{\alpha}_{(u)}$ are estimated parameters for u=1. For u=2, the selecitivity was 1.0 for age-0 and 0.0 for all other ages.
Fishery selectivity	$s_{(f),a}$	$s_{(f),a} = \begin{cases} s_{f,0} & \text{was fixed at 0.0} \\ s_{f,1} & \text{was estimated} \\ s_{f,2} & \text{was fixed at 1.0} \\ s_{f,3} & \text{was fixed at 0.35} \\ s_{f,4+} & \text{was fixed at 0.35} \end{cases}$
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	$Z_{a,y}$	$Z_{a,y} = M_a + F_{a,y}$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Abundance at age	$N_{a,y}$	$N_{0,1977} = \frac{\widehat{R}_0(0.8\varsigma h\phi_{init}-0.2\phi_0(1-h))}{(h-0.2)\phi_{init}} \exp(\widehat{R}_y)$ $\widehat{N}_{1+,1977} \exp(\widehat{N}_a)$ equilibrium conditions expected given assumptions about initial fishing mortality (described below), which includes an estimated deviation from the equilibrium age structure for each age $N_{0,y+1} = \frac{0.8\widehat{R}_0 h S_y}{0.2\phi_0\widehat{R}_0(1-h)+(h-0.2)S_y} \exp(\widehat{R}_{y+1})$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (0 \dots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ $\widehat{R}_0$ (asymptotic maximum recruitment) is an estimated parameter of the spawner-recruit curve, and $\widehat{R}_y$ are estimated annual recruitment deviations in log space for 1977-2011. The bias correction is $\varsigma = \exp(\sigma_R^2/2)$, where σ_R^2 was fixed at 0.6 and was the variance of recruitment deviations. In the SEDAR-32A base run, h=0.75 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 to the gill net index 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 January 1 to correspond with peak spawning $N''_{a,y} = \exp(-t_{\text{spawn}}Z_{a,y})N_{a,y}$
Unfished abundance at age per recruit at time of spawning	NPR_a	$NPR_1 = 1 \times \exp(-t_{\text{spawn}}M_1)$ $NPR_{a+1} = NPR_a \exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (1 \dots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_A t_{\text{spawn}})]}{1-\exp(-M_A)}$
Initial abundance at age per recruit at time of spawning	NPR_a^{init}	Same calculations as for NPR_a, but including fishing mortality (see Z^{init} below).
Unfished spawning biomass per recruit	ϕ_0	$\phi_0 = \sum_{a=0}^{A} NPR_a\rho_a m_a \mathcal{F}_a$ In units of fecundity
Initial spawning biomass per recruit	ϕ_{init}	$\phi_{init} = \sum_{a=0}^{A} NPR_a^{init}\rho_a m_a \mathcal{F}_a$ In units of fecundity
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y}\rho_a m_a \mathcal{F}_a$ Spawning biomass is in units of total fecundity
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 the geometric mean of F from the first three assessment years (1977-1979) and s_a^{init} is the commercial selectivity for these three years.
Initial equilibrium abundance at age	N_a^{eq}	Equilibrium age structure given Z_a^{init}
Population biomass	B_y	$B_y = \sum_a N_{a,y}w_a$
Landings at age in numbers	$L'_{f,a,y}$	$L'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}}N_{a,y}[1-\exp(-Z_{a,y})]$

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Landings at age in weight	$L''_{f,a,y}$	$L''_{f,a,y} = w^L_{f,a,y} L'_{f,a,y}$
Index catchability	$\widehat{q}_u$	estimated constant catchability for each index u
Predicted landings	$\breve{L}_{f,y}$	$\breve{L}_{f,y} = \sum_a L'_{f,a,y}$
Predicted length compositions of fishery independent data	$\breve{p}^{\lambda}_{u,l,y}$	$\breve{p}^{\lambda}_{u,l,y} = \frac{\sum_a \psi_{a,l} s_{u,a,y} N'_{a,y}}{\sum_a s_{u,a,y} N'_{a,y}}$
Predicted age compositions of fishery	$\breve{p}^{\alpha}_{(f),a,y}$	$\breve{p}^{\alpha}_{(f),a,y} = \frac{\mathcal{E} L'_{(f),a,y}}{\sum_a L'_{(f),a,y}}$ this formulation accounts for ageing error
Predicted CPUE	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \widehat{q}_u \sum_a N'_{a,y} s_{u,a}$ where $s_{u,a}$ is the selectivity of index u in the year corresponding to y.

Table 2.1. (continued)

Quantity	Symbol	Description or definition
Objective Function		
Robust multinomial length compositions	Λ_1	$\Lambda_1 = \sum_u \sum_y 0.5 \log(E') - \log\left[\exp\left(-\frac{(p^{\lambda}_{(u),l,y} - \breve{p}^{\lambda}_{(u),l,y})^2}{2E' / (n^{\lambda}_{(u),y}\omega^{\lambda}_{(u)})}\right) + x\right]$ where $E' = \left[(1 - p^{\lambda}_{(u),l,y})(p^{\lambda}_{(u),l,y}) + \frac{0.1}{mbin}\right]$, $mbin$ is the number of length bins, $\omega^{\lambda}_{(u)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero. Bins are 10 mm wide.
Robust multinomial age compositions	Λ_2	$\Lambda_2 = \sum_f \sum_y 0.5 \log(E') - \log\left[\exp\left(-\frac{(p^{\alpha}_{(f),a,y} - \breve{p}^{\alpha}_{(f),a,y})^2}{2E' / (n^{\alpha}_{(f),y}\omega^{\alpha}_{(f)})}\right) + x\right]$ where $E' = \left[(1 - p^{\alpha}_{(f),a,y})(p^{\alpha}_{(f),a,y}) + \frac{0.1}{mbin}\right]$, $mbin$ is the number of age bins, $\omega^{\alpha}_{(f)}$ is a preset weight (selected by iterative re-weighting) and x =1e-5 is an arbitrary value to avoid log zero.
Lognormal landings	Λ_3	$\Lambda_3 = \sum_f \sum_y \frac{\left[\log\left((L_{f,y}+x)/(\breve{L}_{f,y}+x)\right)\right]^2}{2(\sigma^L_{f,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^L_{f,y} = \sqrt{\log(1 + (c^L_{f,y}/\omega^L_f)^2)}$, with $\omega^L_f = 1$ a preset weight.
Lognormal CPUE	Λ_4	$\Lambda_4 = \sum_u \sum_y \frac{\left[\log\left((U_{u,y}+x)/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma^U_{u,y})^2}$ where x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma^U_{u,y} = \sqrt{\log(1 + (c^U_{u,y}/\omega^U_u)^2)}$, with ω^U_u a preset weight.
Lognormal recruitment deviations	Λ_5	$\Lambda_5 = \omega_5 \left[\frac{[R_{1977} + (\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + \sum_{y>1978}^{2011} \frac{[(R_y - \widehat{\varrho} R_{y-1}) + (\widehat{\sigma}^2_R/2)]^2}{2\widehat{\sigma}^2_R} + n \log(\widehat{\sigma}_R)\right]$ where R_y are recruitment deviations in log space, n is the number of years, $\omega_5 = 1$ is a preset weight, $\widehat{\varrho}$ is the first-order autocorrelation, and $\widehat{\sigma}^2_R$ is the estimated recruitment variance ($\varrho = 0$ in the SEDAR-32A base run).
Total objective function	Λ	$\Lambda = \sum_{i=1}^{5} \Lambda_i$ Objective function minimized by the assessment model

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

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR 32A Gulf menhaden assessment August 2013
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
DATA_SECTION

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

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

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

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

// ending year for first selectivity period
init_int endyr_period1;
init_int endyr_period2;

//Total number of ages
init_int nages;

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

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

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

!!cout << "max_F_spr_msy" << max_F_spr_msy  << endl;

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

//###################Commercial Reduction fishery ########################

// Landings  (1000 mt)
init_int styr_cR_L;
init_int endyr_cR_L;
init_vector obs_cR_L(styr_cR_L,endyr_cR_L);   //vector of observed landings by year
init_vector cR_L_cv(styr_cR_L,endyr_cR_L);    //vector of CV of landings by year

// Age Compositions
init_int nyr_cR_agec;
init_ivector yrs_cR_agec(1,nyr_cR_agec);
init_vector nsamp_cR_agec(1,nyr_cR_agec);
init_vector nfish_cR_agec(1,nyr_cR_agec);
init_matrix obs_cR_agec(1,nyr_cR_agec,1,nages);
```

```
//###################### LA gill net index ################################

//CPUE
 init_int styr_gill_cpue;
 init_int endyr_gill_cpue;
 init_vector obs_gill_cpue(styr_gill_cpue,endyr_gill_cpue);
 init_vector gill_cpue_cv(styr_gill_cpue,endyr_gill_cpue);

// Length Compositions (3 cm bins)
init_int nyr_gill_lenc;
init_ivector yrs_gill_lenc(1,nyr_gill_lenc);
init_vector nsamp_gill_lenc(1,nyr_gill_lenc);
init_vector nfish_gill_lenc(1,nyr_gill_lenc);
init_matrix obs_gill_lenc(1,nyr_gill_lenc,1,nlenbins);

!!cout << "start year gill net index" << styr_gill_cpue  << endl;

//###################### seine survey index ##############################
//CPUE
 init_int styr_seine_cpue;
 init_int endyr_seine_cpue;
 init_vector obs_seine_cpue(styr_seine_cpue,endyr_seine_cpue);
 init_vector seine_cpue_cv(styr_seine_cpue,endyr_seine_cpue);



//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: parameter section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##################Single Parameter values and initial guesses ################################
// Von Bert parameters in TL mm all fish
init_vector set_Linf(1,7);
init_vector set_K(1,7);
init_vector set_t0(1,7);
//CV of length at age and its standard error all fish
init_vector set_len_cv(1,7);
//age-independent: used only for MSST and to scale age dependent M, prior if M is estimated
init_vector set_M_constant(1,7);
//Spawner-recruit parameters (Initial guesses or fixed values)
init_vector set_steep(1,7);         //recruitment steepness
init_vector set_log_R0(1,7);        //recruitment R0
init_vector set_R_autocorr(1,7);    //recruitment autocorrelation
init_vector set_rec_sigma(1,7);     //recruitment standard deviation in log space
//Initial guesses or fixed values of estimated selectivity parameters
//reduction fishery
init_vector set_selpar_L50_cR(1,7);
init_vector set_selpar_slope_cR(1,7);
init_vector set_selpar_L502_cR(1,7);   //for double logistic selectivity
init_vector set_selpar_slope2_cR(1,7); //for double logistic selectivity

init_vector set_sel_age0_cR(1,7); //input in logit space by age
init_vector set_sel_age1_cR(1,7);
init_vector set_sel_age2_cR(1,7);
init_vector set_sel_age3_cR(1,7);
init_vector set_sel_age4_cR(1,7);

init_vector set_sel_age0_cR2(1,7); //input in logit space by age-period 2
init_vector set_sel_age1_cR2(1,7);
init_vector set_sel_age2_cR2(1,7);
init_vector set_sel_age3_cR2(1,7);
init_vector set_sel_age4_cR2(1,7);

//gill net survey
init_vector set_selpar_L50_gill(1,7);
init_vector set_selpar_slope_gill(1,7);

//////--index catchability-----------------------------------------------------------------------------------
init_vector set_log_q_gill(1,7);    //catchability coefficient (log) for gill net index
init_vector set_log_q_seine(1,7);      //catchability coefficient (log) SC charterboat logbook cpue index

////--F's-------------------------------
init_vector set_log_avg_F_cR(1,7);

//##################Dev Vector Parameter values (vals) and bounds ################################
//--F vectors--------------------------
init_vector set_log_F_dev_cR(1,3);
init_vector set_log_rec_dev(1,3);
init_vector set_M_dev(1,3);
init_vector set_log_N_dev(1,3);

init_vector set_log_F_dev_cR_vals(styr_cR_L,endyr_cR_L);
init_vector set_log_rec_dev_vals(styr_rec_dev,endyr_rec_dev);
init_vector set_M_dev_vals(styr_seine_cpue,endyr_seine_cpue);
init_vector set_log_N_dev_vals(2,nages);


!!cout << "set_log_rec_dev" << set_log_rec_dev << endl;
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: likelihood weights section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

init_number set_w_L;
init_number set_w_lc_gill;
init_number set_w_ac_cR;
init_number set_w_I_gill;          //weight for gill net index
init_number set_w_I_seine;         //weight for seine index
init_number set_w_M_dev;           //weight on M dev constraint
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

!!cout << "set_w_I_seine" << set_w_I_seine  << endl;
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: miscellaneous stuff section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

//FL(mm)-weight(whole weight in g) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;

//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age
init_vector fec_at_age(1,nages);                //fecundity at age

//weights at start and middle of the year
init_vector wgt_spawn(1,nages);      //weights based on cR fishery
init_vector wgt_start(1,nages);      //weights based on cR fishery

init_number spawn_time_frac; //time of year of peak spawning, as a fraction of the year

// Natural mortality
init_vector set_M(1,nages);     //age-dependent: used in model
init_number max_obs_age;        //max observed age, used to scale M

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

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

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

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

//threshold sample sizes for length comps
init_number minSS_gill_lenc;

//threshold sample sizes for age comps
init_number minSS_cR_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;
int quant_whole;

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)
   {
       cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
       exit(0);  //KWS
   }
   else
   {
    cout << "Data File read correctly" << endl;
   }
 END_CALCS


PARAMETER_SECTION

 LOCAL_CALCS
  const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
  const double K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
  const double t0_LO=set_t0(2); const double t0_HI=set_t0(3); const double t0_PH=set_t0(4);
  const double len_cv_LO=set_len_cv(2); const double len_cv_HI=set_len_cv(3); const double len_cv_PH=set_len_cv(4);
  const double M_constant_LO=set_M_constant(2); const double M_constant_HI=set_M_constant(3); const double M_constant_PH=set_M_constant(4);
  const double steep_LO=set_steep(2); const double steep_HI=set_steep(3); const double steep_PH=set_steep(4);
  const double log_R0_LO=set_log_R0(2); const double log_R0_HI=set_log_R0(3); const double log_R0_PH=set_log_R0(4);
  const double R_autocorr_LO=set_R_autocorr(2); const double R_autocorr_HI=set_R_autocorr(3); const double R_autocorr_PH=set_R_autocorr(4);
  const double rec_sigma_LO=set_rec_sigma(2); const double rec_sigma_HI=set_rec_sigma(3); const double rec_sigma_PH=set_rec_sigma(4);
  const double selpar_L50_cR_LO=set_selpar_L50_cR(2); const double selpar_L50_cR_HI=set_selpar_L50_cR(3); const double selpar_L50_cR_PH=set_selpar_L50_cR(4);
  const double selpar_slope_cR_LO=set_selpar_slope_cR(2); const double selpar_slope_cR_HI=set_selpar_slope_cR(3); const double selpar_slope_cR_PH=set_selpar_slope_cR(4);
  const double selpar_L502_cR_LO=set_selpar_L502_cR(2); const double selpar_L502_cR_HI=set_selpar_L502_cR(3); const double selpar_L502_cR_PH=set_selpar_L502_cR(4);
  const double selpar_slope2_cR_LO=set_selpar_slope2_cR(2); const double selpar_slope2_cR_HI=set_selpar_slope2_cR(3); const double selpar_slope2_cR_PH=set_selpar_slope2_cR(4);
```

```
  const double selpar_age0_cR_LO=set_sel_age0_cR(2); const double selpar_age0_cR_HI=set_sel_age0_cR(3); const double selpar_age0_cR_PH=set_sel_age0_cR(4);
  const double selpar_age1_cR_LO=set_sel_age1_cR(2); const double selpar_age1_cR_HI=set_sel_age1_cR(3); const double selpar_age1_cR_PH=set_sel_age1_cR(4);
  const double selpar_age2_cR_LO=set_sel_age2_cR(2); const double selpar_age2_cR_HI=set_sel_age2_cR(3); const double selpar_age2_cR_PH=set_sel_age2_cR(4);
  const double selpar_age3_cR_LO=set_sel_age3_cR(2); const double selpar_age3_cR_HI=set_sel_age3_cR(3); const double selpar_age3_cR_PH=set_sel_age3_cR(4);
  const double selpar_age4_cR_LO=set_sel_age4_cR(2); const double selpar_age4_cR_HI=set_sel_age4_cR(3); const double selpar_age4_cR_PH=set_sel_age4_cR(4);
  const double selpar_age0_cR2_LO=set_sel_age0_cR2(2); const double selpar_age0_cR2_HI=set_sel_age0_cR2(3); const double selpar_age0_cR2_PH=set_sel_age0_cR2(4);
  const double selpar_age1_cR2_LO=set_sel_age1_cR2(2); const double selpar_age1_cR2_HI=set_sel_age1_cR2(3); const double selpar_age1_cR2_PH=set_sel_age1_cR2(4);
  const double selpar_age2_cR2_LO=set_sel_age2_cR2(2); const double selpar_age2_cR2_HI=set_sel_age2_cR2(3); const double selpar_age2_cR2_PH=set_sel_age2_cR2(4);
  const double selpar_age3_cR2_LO=set_sel_age3_cR2(2); const double selpar_age3_cR2_HI=set_sel_age3_cR2(3); const double selpar_age3_cR2_PH=set_sel_age3_cR2(4);
  const double selpar_age4_cR2_LO=set_sel_age4_cR2(2); const double selpar_age4_cR2_HI=set_sel_age4_cR2(3); const double selpar_age4_cR2_PH=set_sel_age4_cR2(4);
  const double selpar_L50_gill_LO=set_selpar_L50_gill(2); const double selpar_L50_gill_HI=set_selpar_L50_gill(3); const double selpar_L50_gill_PH=set_selpar_L50_gill(4);
  const double selpar_slope_gill_LO=set_selpar_slope_gill(2); const double selpar_slope_gill_HI=set_selpar_slope_gill(3); const double selpar_slope_gill_PH=set_selpar_slope_gill(4);
  const double log_q_gill_LO=set_log_q_gill(2); const double log_q_gill_HI=set_log_q_gill(3); const double log_q_gill_PH=set_log_q_gill(4);
  const double log_q_seine_LO=set_log_q_seine(2); const double log_q_seine_HI=set_log_q_seine(3); const double log_q_seine_PH=set_log_q_seine(4);
  const double log_avg_F_cR_LO=set_log_avg_F_cR(2); const double log_avg_F_cR_HI=set_log_avg_F_cR(3); const double log_avg_F_cR_PH=set_log_avg_F_cR(4);
  //-dev vectors-----------------------------------------------------------------------------------------------------------
  const double log_F_dev_cR_LO=set_log_F_dev_cR(1); const double log_F_dev_cR_HI=set_log_F_dev_cR(2); const double log_F_dev_cR_PH=set_log_F_dev_cR(3);
  const double log_rec_dev_LO=set_log_rec_dev(1); const double log_rec_dev_HI=set_log_rec_dev(2); const double log_rec_dev_PH=set_log_rec_dev(3);
  const double M_dev_LO=set_M_dev(1);const double M_dev_HI=set_M_dev(2);const double M_dev_PH=set_M_dev(3);
  const double N_dev_LO=set_log_N_dev(1);const double N_dev_HI=set_log_N_dev(2);const double N_dev_PH=set_log_N_dev(3);

 END_CALCS

////--------------Growth-------------------------------------------------------------------------
  init_bounded_number Linf(Linf_LO,Linf_HI,Linf_PH);
  init_bounded_number K(K_LO,K_HI,K_PH);
  init_bounded_number t0(t0_LO,t0_HI,t0_PH);
  init_bounded_number len_cv_val(len_cv_LO,len_cv_HI,len_cv_PH);
  vector Linf_out(1,8);
  vector K_out(1,8);
  vector t0_out(1,8);
  vector len_cv_val_out(1,8);

  vector meanlen_FL(1,nages);   //mean fork length (mm) at age all fish
  vector wgt_fish_mt(1,nages);  //wgt in mt
  vector wgt_spawn_mt(1,nages); //wgt in mt

  matrix wholewgt_cR_mt(styr,endyr,1,nages);    //whole wgt of cR landings in mt

  vector lbins(1,nlenbins);

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

  //matrices below are used to match length comps
  matrix lenprob_gill(1,nages,1,nlenbins);     //distn of size at age in gill nets

  //matrices below pertain to the popn at large, used to compute mean weights
  matrix lenprob_gill_all(1,nages,1,nlenbins_all);     //distn of size at age in gill

//  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
//  number log_len_cv
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

////----Predicted length and age compositions
  matrix pred_gill_lenc(1,nyr_gill_lenc,1,nlenbins);
  matrix pred_cR_agec(1,nyr_cR_agec,1,nages);
  matrix ErrorFree_cR_agec(1,nyr_cR_agec,1,nages);


//effective sample size applied in multinomial distributions
  vector nsamp_gill_lenc_allyr(styr,endyr);
  vector nsamp_cR_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_gill_lenc_allyr(styr,endyr);
  vector nfish_cR_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_gill_lenc_allyr_out(styr,endyr);
  vector neff_cR_agec_allyr_out(styr,endyr);

//-----Population-----------------------------------------------------------------------------------
  matrix N(styr,endyr+1,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and cpue
  matrix N_spawn(styr,endyr,1,nages);       //Population numbers by year and age at peaking spawning: used for SSB
  //vector log_Nage_dev(2,nages);
  init_bounded_dev_vector log_Nage_dev(2,nages,N_dev_LO,N_dev_HI,N_dev_PH);
  vector log_Nage_dev_output(2,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+1);                   //Total spawning biomass by year (fecundity in mature ova)
  vector rec(styr,endyr+1);                 //Recruits by year
  vector pred_SPR(styr,endyr);              //spawning biomass-per-recruit (lagged) for Fmed calcs
  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 (fecundity)
  matrix SSBatage(styr,endyr,1,nages);

////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(log_R0_LO,log_R0_HI,log_R0_PH);   //log(virgin Recruitment)
  vector log_R0_out(1,8);
  number R0;                //virgin recruitment

  init_bounded_number steep(steep_LO,steep_HI,steep_PH); //steepness
  vector steep_out(1,8);
  init_bounded_number rec_sigma(rec_sigma_LO,rec_sigma_HI,rec_sigma_PH);  //sd recruitment residuals
  vector rec_sigma_out(1,8);

  number rec_sigma_sq;                      //square of rec_sigma
```

```
  number rec_logL_add;                        //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr_rec_dev,log_rec_dev_LO,log_rec_dev_HI,log_rec_dev_PH);  //log recruitment deviations
  vector log_rec_dev_output(styr,endyr+1);          //used in output. equals zero except for yrs in log_rec_dev

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

  number BiasCor;                                //Bias correction in equilibrium recruits
  init_bounded_number R_autocorr(R_autocorr_LO,R_autocorr_HI,R_autocorr_PH);
  vector R_autocorr_out(1,8);

  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+1);                       //SSB / virgin SSB


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

//Commercial Reduction-----------------------------------------------------
  matrix sel_cR(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cR(selpar_L50_cR_LO,selpar_L50_cR_HI,selpar_L50_cR_PH);
  init_bounded_number selpar_slope_cR(selpar_slope_cR_LO,selpar_slope_cR_HI,selpar_slope_cR_PH);
  init_bounded_number selpar_L502_cR(selpar_L502_cR_LO,selpar_L502_cR_HI,selpar_L502_cR_PH);
  init_bounded_number selpar_slope2_cR(selpar_slope2_cR_LO,selpar_slope2_cR_HI,selpar_slope2_cR_PH);
  vector selpar_L50_cR_out(1,8);
  vector selpar_slope_cR_out(1,8);
  vector selpar_L502_cR_out(1,8);
  vector selpar_slope2_cR_out(1,8);

  init_bounded_number sel_age0_cR_logit(selpar_age0_cR_LO,selpar_age0_cR_HI,selpar_age0_cR_PH);  //cR selectivity at age in logit space
  init_bounded_number sel_age1_cR_logit(selpar_age1_cR_LO,selpar_age1_cR_HI,selpar_age1_cR_PH);
  init_bounded_number sel_age2_cR_logit(selpar_age2_cR_LO,selpar_age2_cR_HI,selpar_age2_cR_PH);
  init_bounded_number sel_age3_cR_logit(selpar_age3_cR_LO,selpar_age3_cR_HI,selpar_age3_cR_PH);
  init_bounded_number sel_age4_cR_logit(selpar_age4_cR_LO,selpar_age4_cR_HI,selpar_age4_cR_PH);
  vector sel_age_cR_vec(1,nages);
  number selpar_age0_cR;
  number selpar_age1_cR;
  number selpar_age2_cR;
  number selpar_age3_cR;
  number selpar_age4_cR;
  vector selpar_age0_cR_out(1,8);
  vector selpar_age1_cR_out(1,8);
  vector selpar_age2_cR_out(1,8);
  vector selpar_age3_cR_out(1,8);
  vector selpar_age4_cR_out(1,8);

  init_bounded_number sel_age0_cR2_logit(selpar_age0_cR2_LO,selpar_age0_cR2_HI,selpar_age0_cR2_PH);  //cR selectivity at age in logit space-period 2
  init_bounded_number sel_age1_cR2_logit(selpar_age1_cR2_LO,selpar_age1_cR2_HI,selpar_age1_cR2_PH);
  init_bounded_number sel_age2_cR2_logit(selpar_age2_cR2_LO,selpar_age2_cR2_HI,selpar_age2_cR2_PH);
  init_bounded_number sel_age3_cR2_logit(selpar_age3_cR2_LO,selpar_age3_cR2_HI,selpar_age3_cR2_PH);
  init_bounded_number sel_age4_cR2_logit(selpar_age4_cR2_LO,selpar_age4_cR2_HI,selpar_age4_cR2_PH);
  vector sel_age_cR2_vec(1,nages);
  number selpar_age0_cR2;
  number selpar_age1_cR2;
  number selpar_age2_cR2;
  number selpar_age3_cR2;
  number selpar_age4_cR2;
  vector selpar_age0_cR2_out(1,8);
  vector selpar_age1_cR2_out(1,8);
  vector selpar_age2_cR2_out(1,8);
  vector selpar_age3_cR2_out(1,8);
  vector selpar_age4_cR2_out(1,8);

//Gill net survey selectivity
  matrix sel_gill(styr_gill_cpue,endyr_gill_cpue,1,nages);
  init_bounded_number selpar_L50_gill(selpar_L50_gill_LO,selpar_L50_gill_HI,selpar_L50_gill_PH);
  init_bounded_number selpar_slope_gill(selpar_slope_gill_LO,selpar_slope_gill_HI,selpar_slope_gill_PH);
  vector selpar_L50_gill_out(1,8);
  vector selpar_slope_gill_out(1,8);

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

//-------------------------------------------------------------------------------------------------------------------------------------------------
//-------CPUE Predictions-------------------------------
  vector pred_gill_cpue(styr_gill_cpue,endyr_gill_cpue);     //predicted gill net U
  matrix N_gill(styr_gill_cpue,endyr_gill_cpue,1,nages);     //used to compute gill net index
  vector pred_seine_cpue(styr_seine_cpue,endyr_seine_cpue);  //predicted seine index
  vector N_seine(styr_seine_cpue,endyr_seine_cpue);          //used to compute seine index

//---Catchability (CPUE q's)-------------------------------------------------------------
  init_bounded_number log_q_gill(log_q_gill_LO,log_q_gill_HI,log_q_gill_PH);
  init_bounded_number log_q_seine(log_q_seine_LO,log_q_seine_HI,log_q_seine_PH);
  vector log_q_gill_out(1,8);
  vector log_q_seine_out(1,8);

  //init_bounded_number q_rate(0.001,0.1,set_q_rate_phase);
  number q_rate;
  vector q_rate_fcn_gill(styr_gill_cpue,endyr_gill_cpue);     //increase due to technology creep (saturates in 2003)
  vector q_rate_fcn_seine(styr_seine_cpue,endyr_seine_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

  vector q_RW_log_dev_gill(styr_gill_cpue,endyr_gill_cpue-1);
```

```
  vector q_RW_log_dev_seine(styr_seine_cpue,endyr_seine_cpue-1);

  vector q_gill(styr_gill_cpue,endyr_gill_cpue); //number q_gill;
  vector q_seine(styr_seine_cpue,endyr_seine_cpue); //number q_seine;

//-------------------------------------------------------------------------------------------------------------------------------------------
//---Landings in numbers (total or 1000 fish) and in wgt (1000s mt)-------------------------------------------------
  matrix L_cR_num(styr,endyr,1,nages);  //landings (numbers) at age
  matrix L_cR_mt(styr,endyr,1,nages);   //landings (mt) at age
  vector pred_cR_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_cR_L_mt(styr,endyr);     //yearly landings in 1000s mt summed over ages

  matrix L_total_num(styr,endyr,1,nages); //total landings in number at age
  matrix L_total_mt(styr,endyr,1,nages);  //landings in mt at age
  vector L_total_knum_yr(styr,endyr);     //total landings in 1000 fish by yr summed over ages
  vector L_total_mt_yr(styr,endyr);      //total landings (1000s mt) by yr summed over ages

////---MSY calcs-------------------------------------------------------------------------
  number F_cR_prop;       //proportion of F_sum attributable to cR
  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 fecundity) at msy
  number F_msy_out;             //F at msy
  number msy_mt_out;            //max sustainable yield (1000s mt)
  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_mt(1,n_iter_msy);      //equilibrium landings(1000s mt) 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 (fecundity) values corresponding to F values in F_msy
  vector B_eq(1,n_iter_msy);         //equilibrium biomass values corresponding to F values in F_msy

  vector FdF_msy(styr,endyr);
  vector SdSSB_msy(styr,endyr+1);
  number SdSSB_msy_end;
  number FdF_msy_end;
  number FdF_msy_end_mean;          //geometric mean of last 3 yrs

  vector wgt_wgted_L_mt(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)


////---Fmed calcs-------------------------------------------------------------------------
  number quant_decimal;
  number quant_diff;
  number quant_result;

  number R_med;                                 //median recruitment for chosen benchmark years
  vector R_temp(styr,endyr);
  vector R_sort(styr,endyr);
  number SPR_med;                               //median SSB/R (R = SSB year+1) for chosen SSB years
  number SPR_75th;
  vector SPR_temp(styr,endyr);
  vector SPR_sort(styr,endyr);
  number SSB_med;                               //SSB corresponding to SSB/R median and R median
  number SSB_med_thresh;                        //SSB threshold
  vector SPR_diff(1,n_iter_spr);
  number SPR_diff_min;
  number F_med;                                 //Fmed benchmark
  number F_med_target;
  number F_med_age2plus;                        //Fmed benchmark
  number F_med_target_age2plus;
  number L_med;
  number L_med_target;


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

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

  vector M(1,nages);                  //age-dependent natural mortality
  init_bounded_number M_constant(M_constant_LO,M_constant_HI,M_constant_PH);   //age-indpendent: used only for MSST
  vector M_constant_out(1,8);
//---------------set up for M at age-1 to be estimated
  init_bounded_dev_vector M_dev(styr_seine_cpue,endyr_seine_cpue,M_dev_LO,M_dev_HI,M_dev_PH);  //M devs deviations
  vector M_dev_output(styr_seine_cpue,endyr_seine_cpue);

  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_cR(log_avg_F_cR_LO,log_avg_F_cR_HI,log_avg_F_cR_PH);
  vector log_avg_F_cR_out(1,8);
  init_bounded_dev_vector log_F_dev_cR(styr_cR_L,endyr_cR_L,log_F_dev_cR_LO,log_F_dev_cR_HI,log_F_dev_cR_PH);
  vector log_F_dev_cR_out(styr_cR_L,endyr_cR_L);
  matrix F_cR(styr,endyr,1,nages);
  vector F_cR_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cR;
  number log_F_dev_end_cR;

  vector sel_initial(1,nages);       //initial selectivity (commercial selectivity)

//---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(mt)-per-recruit (ypr) values corresponding to F values in F_spr

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

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

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

  number sdnr_lc_gill;

  number sdnr_ac_cR;

  number sdnr_I_gill;
  number sdnr_I_seine;

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

  number w_lc_gill;

  number w_ac_cR;

  number w_I_gill;
  number w_I_seine;

  number w_M_dev;
  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_gill_cpue;
  number f_seine_cpue;

  number f_cR_L;

  number f_gill_lenc;

  number f_cR_agec;

  number f_gill_RW_cpue;   //random walk component of indices
  number f_seine_RW_cpue;   //random walk component of indices


  //Penalties and constraints. Not all are used.
  number f_M_dev;                  //likelihood component constraint for annual M devs
  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

  objective_function_value fval;
  number fval_data;

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

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


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
GLOBALS_SECTION
  #include "admodel.h"          // Include AD class definitions
```

```
  #include "admb2r.cpp"    // Include S-compatible output functions (needs preceding)
  #include <time.h>
time_t start,finish;
long hour,minute,second;
double elapsed_time;

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

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

// Set values of fixed parameters or set initial guess of estimated parameters
  Linf=set_Linf(1);
  K=set_K(1);
  t0=set_t0(1);
  len_cv_val=set_len_cv(1);

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

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

  log_q_gill=set_log_q_gill(1);
  log_q_seine=set_log_q_seine(1);

  q_rate=set_q_rate;
  q_rate_fcn_gill=1.0;
  q_rate_fcn_seine=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;
  q_RW_log_dev_gill.initialize();
  q_RW_log_dev_seine.initialize();


  if (set_q_rate_phase<0 & q_rate!=0.0)
  {

      for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
      {   if (iyear>styr_gill_cpue & iyear <=2003)
          {//q_rate_fcn_gill(iyear)=(1.0+q_rate)*q_rate_fcn_gill(iyear-1); //compound
             q_rate_fcn_gill(iyear)=(1.0+(iyear-styr_gill_cpue)*q_rate)*q_rate_fcn_gill(styr_gill_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_gill(iyear)=q_rate_fcn_gill(iyear-1);}
      }

      for (iyear=styr_seine_cpue; iyear<=endyr_seine_cpue; iyear++)
      {   if (iyear>styr_seine_cpue & iyear <=2003)
          {//q_rate_fcn_seine(iyear)=(1.0+q_rate)*q_rate_fcn_seine(iyear-1); //compound
             q_rate_fcn_seine(iyear)=(1.0+(iyear-styr_seine_cpue)*q_rate)*q_rate_fcn_seine(styr_seine_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_seine(iyear)=q_rate_fcn_seine(iyear-1);}
      }

  } //end q_rate conditional

  w_L=set_w_L;

  w_lc_gill=set_w_lc_gill;

  w_ac_cR=set_w_ac_cR;

  w_I_gill=set_w_I_gill;
  w_I_seine=set_w_I_seine;

  w_M_dev=set_w_M_dev;
  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_cR=set_log_avg_F_cR(1);
  log_F_dev_cR=set_log_F_dev_cR_vals;
  log_Nage_dev=set_log_N_dev_vals;

  selpar_L50_cR=set_selpar_L50_cR(1);
  selpar_slope_cR=set_selpar_slope_cR(1);
  selpar_L502_cR=set_selpar_L502_cR(1);
  selpar_slope2_cR=set_selpar_slope2_cR(1);
  selpar_L50_gill=set_selpar_L50_gill(1);
  selpar_slope_gill=set_selpar_slope_gill(1);

  sel_age0_cR_logit=set_sel_age0_cR(1);  //setting cR selectivity at age in logit space
  sel_age1_cR_logit=set_sel_age1_cR(1);
  sel_age2_cR_logit=set_sel_age2_cR(1);
  sel_age3_cR_logit=set_sel_age3_cR(1);
  sel_age4_cR_logit=set_sel_age4_cR(1);

  sel_age0_cR2_logit=set_sel_age0_cR2(1);  //setting cR selectivity at age in logit space
  sel_age1_cR2_logit=set_sel_age1_cR2(1);
  sel_age2_cR2_logit=set_sel_age2_cR2(1);
  sel_age3_cR2_logit=set_sel_age3_cR2(1);
  sel_age4_cR2_logit=set_sel_age4_cR2(1);
```

```
  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;

  lbins=lenbins;

  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_gill_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nsamp_cR_agec_allyr=missing;

      nfish_gill_lenc_allyr=missing;//"missing" defined in admb2r.cpp
      nfish_cR_agec_allyr=missing;

      for (iyear=1; iyear<=nyr_gill_lenc; iyear++)
         {if (nsamp_gill_lenc(iyear)>=minSS_gill_lenc)
            {nsamp_gill_lenc_allyr(yrs_gill_lenc(iyear))=nsamp_gill_lenc(iyear);
             nfish_gill_lenc_allyr(yrs_gill_lenc(iyear))=nfish_gill_lenc(iyear);}}

      for (iyear=1; iyear<=nyr_cR_agec; iyear++)
         {if (nsamp_cR_agec(iyear)>=minSS_cR_agec)
           {nsamp_cR_agec_allyr(yrs_cR_agec(iyear))=nsamp_cR_agec(iyear);
            nfish_cR_agec_allyr(yrs_cR_agec(iyear))=nfish_cR_agec(iyear);}}


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


//fill in F's, Catch matrices, and log rec dev with zero's
  F_cR.initialize();
  L_cR_num.initialize();

  F_cR_out.initialize();

  sel_cR.initialize();
  sel_gill.initialize();

  log_rec_dev_output.initialize();
  log_Nage_dev_output.initialize();
  log_rec_dev=set_log_rec_dev_vals;

  log_Nage_dev.initialize();
  M_dev_output.initialize();


//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
TOP_OF_MAIN_SECTION
  time(&start);
  arrmblsize=20000000;
  gradient_structure::set_MAX_NVAR_OFFSET(1600);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(2000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(2000000);
  gradient_structure::set_NUM_DEPENDENT_VARIABLES(10000);

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

 R0=mfexp(log_R0);

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

 get_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;
 //exit(0);
 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 FL) and weight (whole) at age
    meanlen_FL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));    //fork length in mm
    wgt_fish_mt=g2mt*wgt_start; //wgt in mt
    wgt_spawn_mt=g2mt*wgt_spawn;   //mt of whole wgt


FUNCTION get_reprod
   //for reproductive capacity calcs
   //product of sex ratio, maturity, and fecundity for gulf menhaden
   reprod=elem_prod(elem_prod(prop_f,maturity_f),fec_at_age);


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_FL(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_FL(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
  lenprob_gill=lenprob;

  lenprob_gill_all=lenprob_all;


FUNCTION get_weight_at_age_landings

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    wholewgt_cR_mt(iyear)=wgt_fish_mt;  //whole weight in mt

  }

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

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

FUNCTION get_selectivity
  selpar_age0_cR=1.0/(1.0+mfexp(-sel_age0_cR_logit));
  selpar_age1_cR=1.0/(1.0+mfexp(-sel_age1_cR_logit));
  //selpar_age2_cR=1.0/(1.0+mfexp(-sel_age2_cR_logit));
  selpar_age2_cR=1.0;
  //selpar_age3_cR=1.0/(1.0+mfexp(-sel_age3_cR_logit));
  selpar_age3_cR=0.35;
  //selpar_age4_cR=1.0/(1.0+mfexp(-sel_age3_cR_logit));
  selpar_age4_cR=0.35;
  sel_age_cR_vec(1)=selpar_age0_cR;
  sel_age_cR_vec(2)=selpar_age1_cR;
  sel_age_cR_vec(3)=selpar_age2_cR;
  sel_age_cR_vec(4)=selpar_age3_cR;
  sel_age_cR_vec(5)=selpar_age4_cR;

  selpar_age0_cR2=1.0/(1.0+mfexp(-sel_age0_cR2_logit));
```

```
  selpar_age1_cR2=1.0/(1.0+mfexp(-sel_age1_cR2_logit));
  //selpar_age2_cR2=1.0/(1.0+mfexp(-sel_age2_cR2_logit));
  selpar_age2_cR2=1.0;
  //selpar_age3_cR2=1.0/(1.0+mfexp(-sel_age3_cR_logit));
  selpar_age3_cR2=0.35;
  //selpar_age4_cR2=1.0/(1.0+mfexp(-sel_age3_cR_logit));
  selpar_age4_cR2=0.35;
  sel_age_cR2_vec(1)=selpar_age0_cR2;
  sel_age_cR2_vec(2)=selpar_age1_cR2;
  sel_age_cR2_vec(3)=selpar_age2_cR2;
  sel_age_cR2_vec(4)=selpar_age3_cR2;
  sel_age_cR2_vec(5)=selpar_age4_cR2;

  for (iyear=styr; iyear<=endyr_period1; iyear++)
  {
     //sel_cR(iyear)=logistic(agebins, selpar_L50_cR, selpar_slope_cR);
     //sel_cR(iyear)=logistic_double(agebins, selpar_L50_cR, selpar_slope_cR, selpar_L502_cR, selpar_slope2_cR);
     sel_cR(iyear)=sel_age_cR_vec;
  }

  for (iyear=(endyr_period1+1); iyear<=endyr_period2; iyear++)
  {
     //sel_cR(iyear)=logistic(agebins, selpar_L50_cR, selpar_slope_cR);
     //sel_cR(iyear)=logistic_double(agebins, selpar_L50_cR, selpar_slope_cR, selpar_L502_cR, selpar_slope2_cR);
     //sel_cR(iyear)=sel_age_cR2_vec;
     sel_cR(iyear)=sel_age_cR_vec;
  }

  for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
  {
     sel_gill(iyear)=logistic(agebins,selpar_L50_gill,selpar_slope_gill);
  }

  sel_initial=sel_cR(styr);

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();

  //initialization F is avg from first 3 yrs of observed landings
  log_F_dev_init_cR=sum(log_F_dev_cR(styr_cR_L,(styr_cR_L+2)))/3.0;

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    if(iyear>=styr_cR_L & iyear<=endyr_cR_L)
    {  F_cR_out(iyear)=mfexp(log_avg_F_cR+log_F_dev_cR(iyear));
       F_cR(iyear)=sel_cR(iyear)*F_cR_out(iyear);
       Fsum(iyear)+=F_cR_out(iyear);
    }

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

    Fapex(iyear)=max(F(iyear));
    Z(iyear)=M+F(iyear);

    if(iyear>=styr_seine_cpue & iyear<=endyr_seine_cpue)
    { Z(iyear,2)=M(2)+M_dev(iyear)+F(iyear,2); //adds deviations in age-1 M
    }

  }  //end iyear


FUNCTION get_bias_corr
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_dev)-
              sum(log_rec_dev(styr_rec_dev,endyr_rec_dev))/nyrs_rec)
              /(nyrs_rec-1.0);

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

FUNCTION get_numbers_at_age
//Initialization

  S0=spr_F0*R0;  //virgin SSB

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

  B0=bpr_F0*R_virgin*1000000;  //virgin biomass
  B0_q_DD=R_virgin*sum(elem_prod(N_bpr_F0(set_q_DD_stage,nages),wgt_fish_mt(set_q_DD_stage,nages)));

  F_initial=sel_cR(styr)*mfexp(log_avg_F_cR+log_F_dev_init_cR);
  Z_initial=M+F_initial;

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

  spr_initial=sum(elem_prod(N_spr_initial,reprod)); //initial ssb for s-r curve

  R1=SR_eq_func(R0, steep, spr_F0, spr_initial, BiasCor, SR_switch)*mfexp(log_rec_dev(styr_rec_dev));
  //R1=SR_eq_func(R0, steep, spr_F0, spr_initial, BiasCor, SR_switch);
  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(styr,1)=N_initial_eq(1);

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

  SSB(styr)=sum(elem_prod(N_spawn(styr),reprod));
  B_q_DD(styr)=sum(elem_prod(N(styr)(set_q_DD_stage,nages),wgt_fish_mt(set_q_DD_stage,nages)));


//Rest of years
  for (iyear=styr; iyear<endyr; iyear++)
  {
    if(iyear<(styr_rec_dev-1)||iyear>(endyr_rec_dev-1)) //recruitment follows S-R curve exactly
    {
        //N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch);
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        //N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5)));
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_fish_mt(set_q_DD_stage,nages)));

        N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear+1),SR_switch);
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5)));
    }
    else   //recruitment follows S-R curve with lognormal deviation
    {
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_fish_mt(set_q_DD_stage,nages)));

        N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear+1),SR_switch)*mfexp(log_rec_dev(iyear+1));
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5)));
    }
  }

    //values for projections
    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));
    N(endyr+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(endyr+1),SR_switch);

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

 for (iyear=styr; iyear<=endyr; iyear++)
  {
    pred_SPR(iyear)=SSB(iyear)/rec(iyear);
  }

FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_cR_num(iyear,iage)=N(iyear,iage)*F_cR(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);

    }
    pred_cR_L_knum(iyear)=sum(L_cR_num(iyear));//landings already being estimated in 1000s

  }


FUNCTION get_landings_wgt

////---Predicted landings-----------------------
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    L_cR_mt(iyear)=elem_prod(L_cR_num(iyear),wholewgt_cR_mt(iyear))*1000000; //in 1000 mt

    pred_cR_L_mt(iyear)=sum(L_cR_mt(iyear));
  }


FUNCTION get_catchability_fcns
 //Get rate increase if estimated, otherwise fixed above
  if (set_q_rate_phase>0.0)
  {

      for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
      {   if (iyear>styr_gill_cpue & iyear <=2003)
          {//q_rate_fcn_gill(iyear)=(1.0+q_rate)*q_rate_fcn_gill(iyear-1); //compound
             q_rate_fcn_gill(iyear)=(1.0+(iyear-styr_gill_cpue)*q_rate)*q_rate_fcn_gill(styr_gill_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_gill(iyear)=q_rate_fcn_gill(iyear-1);}
      }
```

```
      for (iyear=styr_seine_cpue; iyear<=endyr_seine_cpue; iyear++)
      {   if (iyear>styr_seine_cpue & iyear <=2003)
          {//q_rate_fcn_seine(iyear)=(1.0+q_rate)*q_rate_fcn_seine(iyear-1); //compound
             q_rate_fcn_seine(iyear)=(1.0+(iyear-styr_seine_cpue)*q_rate)*q_rate_fcn_seine(styr_seine_cpue);  //linear
          }
          if (iyear>2003) {q_rate_fcn_seine(iyear)=q_rate_fcn_seine(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------------------------

  //Gill net index
  q_gill(styr_gill_cpue)=mfexp(log_q_gill);
  for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
  {   N_gill(iyear)=elem_prod(N_mdyr(iyear),sel_gill(iyear));
      pred_gill_cpue(iyear)=q_gill(iyear)*q_rate_fcn_gill(iyear)*q_DD_fcn(iyear)*sum(N_gill(iyear));
      if (iyear<endyr_gill_cpue){q_gill(iyear+1)=q_gill(iyear)*mfexp(q_RW_log_dev_gill(iyear));}
  }

 //seine index
  q_seine(styr_seine_cpue)=mfexp(log_q_seine);
  for (iyear=styr_seine_cpue; iyear<=endyr_seine_cpue; iyear++)
  {   N_seine(iyear)=N(iyear,1)*mfexp(-1.*(Z(iyear)(1)*0.25));//matching seine index with April 1 (1/4 of the year completed)
      pred_seine_cpue(iyear)=q_seine(iyear)*q_rate_fcn_seine(iyear)*q_DD_fcn(iyear)*N_seine(iyear);
      if (iyear<endyr_seine_cpue){q_seine(iyear+1)=q_seine(iyear)*mfexp(q_RW_log_dev_seine(iyear));}
  }


FUNCTION get_length_comps

 //gill net survey
  for (iyear=1;iyear<=nyr_gill_lenc;iyear++)
  {
        pred_gill_lenc(iyear)=(N_gill(yrs_gill_lenc(iyear))*lenprob_gill)
                        /sum(N_gill(yrs_gill_lenc(iyear)));
  }


FUNCTION get_age_comps

 //Commerical reduction
 for (iyear=1;iyear<=nyr_cR_agec;iyear++)
  {
    ErrorFree_cR_agec(iyear)=L_cR_num(yrs_cR_agec(iyear))/sum(L_cR_num(yrs_cR_agec(iyear)));
    pred_cR_agec(iyear)=age_error*ErrorFree_cR_agec(iyear);
  }



////-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cR+
        sum(log_F_dev_cR((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);

  F_cR_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cR+
        sum(log_F_dev_cR((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;

  log_F_dev_end_cR=sum(log_F_dev_cR((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

  F_end_L=sel_cR(endyr)*mfexp(log_avg_F_cR+log_F_dev_end_cR);

  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_cR_prop;
  wgt_wgted_L_mt=F_cR_prop/wgt_wgted_L_denom*wholewgt_cR_mt(endyr)*1000; //to scale to 1000s mt

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


    if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
    N_age_msy*=R_eq(ff);
    N_age_msy_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_spawn_mt))*1000000;//to scale to 1000s mt and catch in 1000s
    L_eq_mt(ff)=sum(elem_prod(L_age_msy,wgt_wgted_L_mt))*1000;//to scale to catch in 1000s, wgt_wgted_L_mt is already scaled to 1000s mt
    L_eq_knum(ff)=sum(L_age_msy)/1000.0;
  }

  msy_mt_out=max(L_eq_mt);

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

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

//switch here if var_rec_dev <=dzero
  if(var_rec_dev>0.0)
   {sigma_rec_dev=sqrt(var_rec_dev);} //pow(var_rec_dev,0.5);  //sample SD of predicted residuals (may not equal rec_sigma)
   else{sigma_rec_dev=0.0;}

  len_cv=elem_div(len_sd,meanlen_FL);

  //compute total landings-at-age in 1000 fish and 1000s mt
  L_total_num.initialize();
  L_total_mt.initialize();
  L_total_knum_yr.initialize();
  L_total_mt_yr.initialize();

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_mt_yr(iyear)=pred_cR_L_mt(iyear);
        L_total_knum_yr(iyear)=pred_cR_L_knum(iyear);

        B(iyear)=elem_prod(N(iyear),wgt_spawn_mt)*1000000;//scale to 1000s mt and 1000s fish landed
        totN(iyear)=sum(N(iyear));  //in 1000s of fish
        totB(iyear)=sum(B(iyear));  //in 1000s of mt
        SSBatage(iyear)=elem_prod(N(iyear),reprod);
  }

  L_total_num=L_cR_num;   //landings at age in 1000s fish
  L_total_mt=L_cR_mt;   //landings at age in 1000s mt whole weight

  B(endyr+1)=elem_prod(N(endyr+1),wgt_spawn_mt)*1000000;//scale to 1000s mt and 1000s fish
  totN(endyr+1)=sum(N(endyr+1));//in 1000s of fish
  totB(endyr+1)=sum(B(endyr+1));//in 1000s of mt


  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);}

   //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_mt(iage)*1000; //already scaled to 1000s mt, but need to scale to 1000s fish
    }
  }

FUNCTION get_effective_sample_sizes

       neff_gill_lenc_allyr_out=missing;//"missing" defined in admb2r.cpp
       neff_cR_agec_allyr_out=missing;


       for (iyear=1; iyear<=nyr_gill_lenc; iyear++)
          {if (nsamp_gill_lenc(iyear)>=minSS_gill_lenc)
             {neff_gill_lenc_allyr_out(yrs_gill_lenc(iyear))=multinom_eff_N(pred_gill_lenc(iyear),obs_gill_lenc(iyear));}
           else {neff_gill_lenc_allyr_out(yrs_gill_lenc(iyear))=-99;}
          }

       for (iyear=1; iyear<=nyr_cR_agec; iyear++)
          {if (nsamp_cR_agec(iyear)>=minSS_cR_agec)
             {neff_cR_agec_allyr_out(yrs_cR_agec(iyear))=multinom_eff_N(pred_cR_agec(iyear),obs_cR_agec(iyear));}
           else {neff_cR_agec_allyr_out(yrs_cR_agec(iyear))=-99;}
          }


FUNCTION get_Fmed_benchmarks

  //sorting function for recruitment and SPR values (slow algorithm, but works)
  R_temp=rec(styr,endyr);
  SPR_temp=pred_SPR(styr,endyr);
  for(int jyear=endyr; jyear>=styr; jyear--)
  {
    R_sort(jyear)=max(R_temp);
    SPR_sort(jyear)=max(SPR_temp);
    for(iyear=styr; iyear<=endyr; iyear++)
    {
      if(R_temp(iyear)==R_sort(jyear))
      {
        R_temp(iyear)=0.0;
      }
      if(SPR_temp(iyear)==SPR_sort(jyear))
      {
        SPR_temp(iyear)=0.0;
      }
    }
  }

  // compute the quantile using quant_whole (declared in the data section)
  // which computes the floor integer of a decimal number
  //median
  quant_decimal=(endyr-styr)*0.5;
  quant_whole=(endyr-styr)*0.5;
  quant_diff=quant_decimal-quant_whole;
  R_med=R_sort(styr+quant_whole)*(1-quant_diff)+R_sort(styr+quant_whole+1)*(quant_diff);
  SPR_med=SPR_sort(styr+quant_whole)*(1-quant_diff)+SPR_sort(styr+quant_whole+1)*(quant_diff);
  //cout << "quant_decimal = " << quant_decimal << endl;
  //cout << "quant_whole = " << quant_whole << endl;
```

```
  //cout << "quant_diff = " << quant_diff << endl;
  //cout << "result = " << quant_whole*(1-quant_diff)+(quant_whole+1)*quant_diff << endl;
  //cout << "R_med = " << R_med << endl;
  //cout << "R_sort = " << R_sort << endl;
  //cout << "R = " << R_temp << endl;

  //75th quantile
  quant_decimal=(endyr-styr)*0.75;
  quant_whole=(endyr-styr)*0.75;
  quant_diff=quant_decimal-quant_whole;
  SPR_75th=SPR_sort(styr+quant_whole)*(1-quant_diff)+SPR_sort(styr+quant_whole+1)*(quant_diff);
  //cout << "quant_decimal = " << quant_decimal << endl;
  //cout << "quant_whole = " << quant_whole << endl;
  //cout << "quant_diff = " << quant_diff << endl;
  //cout << "result = " << quant_whole*(1-quant_diff)+(quant_whole+1)*quant_diff << endl;

  //find F that matches SPR_med = F_med
  SPR_diff=square(spr_spr-SPR_med);
  SPR_diff_min=min(SPR_diff);
  for(ff=1; ff<=n_iter_spr; ff++)
  {
    if(SPR_diff(ff)==SPR_diff_min)
    {
      F_med=F_spr(ff);
      //F_med_age2plus=F_spr_age2plus(ff);
      L_med=L_spr(ff)*R_med;
    }
  }
  SSB_med=SPR_med*R_med;
  SSB_med_thresh=SSB_med*0.5;

  //get the target that corresponds to Fmed, based on 75th quantile of SPR scatter
  SPR_diff=square(spr_spr-SPR_75th);
  SPR_diff_min=min(SPR_diff);
  for(ff=1; ff<=n_iter_spr; ff++)
  {
    if(SPR_diff(ff)==SPR_diff_min)
    {
      F_med_target=F_spr(ff);
      //F_med_target_age2plus=F_spr_age2plus(ff);
      L_med_target=L_spr(ff)*R_med;
    }
  }


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


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

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

  f_gill_cpue=0.0;
  f_gill_cpue=lk_lognormal(pred_gill_cpue, obs_gill_cpue, gill_cpue_cv, w_I_gill);
  fval+=f_gill_cpue;
  fval_data+=f_gill_cpue;

  f_seine_cpue=0.0;
  f_seine_cpue=lk_lognormal(pred_seine_cpue, obs_seine_cpue, seine_cpue_cv, w_I_seine);
  fval+=f_seine_cpue;
  fval_data+=f_seine_cpue;

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

  //f_cR_L in 1000s mt
  f_cR_L=lk_lognormal(pred_cR_L_mt, obs_cR_L, cR_L_cv, w_L);

  fval+=f_cR_L;
  fval_data+=f_cR_L;


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

  //f_gill_lenc
  f_gill_lenc=lk_robust_multinomial(nsamp_gill_lenc, pred_gill_lenc, obs_gill_lenc, nyr_gill_lenc, double(nlenbins), minSS_gill_lenc, w_lc_gill);
  //f_gill_lenc=lk_multinomial(nsamp_gill_lenc, pred_gill_lenc, obs_gill_lenc, nyr_gill_lenc, minSS_gill_lenc, w_lc_gill);
  fval+=f_gill_lenc;
  fval_data+=f_gill_lenc;



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

  //f_cR_agec
  f_cR_agec=lk_robust_multinomial(nsamp_cR_agec, pred_cR_agec, obs_cR_agec, nyr_cR_agec, double(nages), minSS_cR_agec, w_ac_cR);
  //f_cR_agec=lk_multinomial(nsamp_cR_agec, pred_cR_agec, obs_cR_agec, nyr_cR_agec, minSS_cR_agec, w_ac_cR);
  fval+=f_cR_agec;
  fval_data+=f_cR_agec;


////-----------Constraints and penalties---------------------------------
  f_M_dev=0.0;
  f_M_dev=norm2(M_dev);
  fval+=w_M_dev*f_M_dev;



  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_rec_dev; iyear++)
    {f_rec_dev+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
               (2.0*rec_sigma_sq));}
  f_rec_dev+=rec_logL_add;
  fval+=w_rec*f_rec_dev;

  f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
  if (w_rec_early>0.0)
    { if (styr_rec_dev<endyr_rec_phase1)
        {
          for(iyear=styr_rec_dev; iyear<=endyr_rec_phase1; iyear++)
          //{f_rec_dev_early+=(square(log_rec_dev(iyear)-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_gill_RW_cpue=0.0;
  //for (iyear=styr_gill_cpue; iyear<endyr_gill_cpue; iyear++)
  //    {f_gill_RW_cpue+=square(q_RW_log_dev_gill(iyear))/(2.0*set_q_RW_gill_var);}
  //fval+=f_gill_RW_cpue;

  //f_seine_RW_cpue=0.0;
  //for (iyear=styr_seine_cpue; iyear<endyr_seine_cpue; iyear++)
  //    {f_seine_RW_cpue+=square(q_RW_log_dev_seine(iyear))/(2.0*set_q_RW_seine_var);}
  //fval+=f_seine_RW_cpue;


//---Priors-----------------------------------------------------
//neg_log_prior arguments: estimate, prior mean, prior var/-CV, pdf type
//Variance input as a negative value is considered to be CV in arithmetic space (CV=-1 implies loose prior)
//pdf type 1=none, 2=lognormal, 3=normal, 4=beta
  f_priors=0.0;
  //f_priors+=neg_log_prior(Linf,set_Linf(5),set_Linf(6),set_Linf(7));
  //f_priors+=neg_log_prior(K,set_K(5),set_K(6),set_K(7));
  //f_priors+=neg_log_prior(t0,set_t0(5),set_t0(6),set_t0(7));
  //f_priors+=neg_log_prior(len_cv_val,set_len_cv(5),set_len_cv(6),set_len_cv(7));
  //f_priors+=neg_log_prior(M_constant,set_M_constant(5),set_M_constant(6),set_M_constant(7));

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

  //f_priors+=neg_log_prior(selpar_L50_cR,set_selpar_L50_cR(5),set_selpar_L50_cR(6), set_selpar_L50_cR(7));
  //f_priors+=neg_log_prior(selpar_slope_cR,set_selpar_slope_cR(5),set_selpar_slope_cR(6), set_selpar_slope_cR(7));
  //f_priors+=neg_log_prior(selpar_L502_cR,set_selpar_L502_cR(5),set_selpar_L502_cR(6), set_selpar_L502_cR(7));
  //f_priors+=neg_log_prior(selpar_slope2_cR,set_selpar_slope2_cR(5),set_selpar_slope2_cR(6), set_selpar_slope2_cR(7));
  //f_priors+=neg_log_prior(sel_age1_cR_logit,set_sel_age1_cR(5),set_sel_age1_cR(6), set_sel_age1_cR(7));
  //f_priors+=neg_log_prior(sel_age3_cR_logit,set_sel_age3_cR(5),set_sel_age3_cR(6), set_sel_age3_cR(7));
  //f_priors+=neg_log_prior(sel_age4_cR_logit,set_sel_age4_cR(5),set_sel_age4_cR(6), set_sel_age4_cR(7));

  //f_priors+=neg_log_prior(selpar_L50_gill,set_selpar_L50_gill(5),set_selpar_L50_gill(6), set_selpar_L50_gill(7));
  //f_priors+=neg_log_prior(selpar_slope_gill,set_selpar_slope_gill(5),set_selpar_slope_gill(6), set_selpar_slope_gill(7));

  //f_priors+=neg_log_prior(log_q_gill,set_log_q_gill(5),set_log_q_gill(6),set_log_q_gill(7));
  //f_priors+=neg_log_prior(log_q_seine,set_log_q_seine(5),set_log_q_seine(6),set_log_q_seine(7));

  //f_priors+=neg_log_prior(log_avg_F_cR,set_log_avg_F_cR(5),set_log_avg_F_cR(6),set_log_avg_F_cR(7));


  fval+=f_priors;

  //cout << "fval = " << fval << "  fval_data = " << fval_data << endl;
  //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 or Ricker)
FUNCTION dvariable SR_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const dvariable& SSB, int func)
  //R0=virgin recruitment, h=steepness, spr_F0=spawners per recruit @ F=0, SSB=spawning biomass
  //func=1 for Beverton-Holt, 2 for Ricker
  RETURN_ARRAYS_INCREMENT();
  dvariable Recruits_Tmp;
  switch(func) {
    case 1: //Beverton-Holt
      Recruits_Tmp=((0.8*R0*h*SSB)/(0.2*R0*spr_F0*(1.0-h)+(h-0.2)*SSB));
    break;
    case 2: //Ricker
      Recruits_Tmp=((SSB/spr_F0)*mfexp(h*(1-SSB/(R0*spr_F0))));
    break;
  }
  RETURN_ARRAYS_DECREMENT();
  return Recruits_Tmp;

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


//--------------------------------------------------------------------------------
//compute multinomial effective sample size for a single yr
FUNCTION dvariable multinom_eff_N(const dvar_vector& pred_comp, const dvar_vector& obs_comp)
  //pred_comp=vector of predicted comps, obscomp=vector of observed comps
  dvariable EffN_Tmp; dvariable numer; dvariable denom;
  RETURN_ARRAYS_INCREMENT();
  numer=sum( elem_prod(pred_comp,(1.0-pred_comp)) );
  denom=sum( square(obs_comp-pred_comp) );
```

```
  if (denom>0.0) {EffN_Tmp=numer/denom;}
  else {EffN_Tmp=-missing;}
  RETURN_ARRAYS_DECREMENT();
  return EffN_Tmp;

//--------------------------------------------------------------------------------
//Likelihood contribution: lognormal
FUNCTION dvariable lk_lognormal(const dvar_vector& pred, const dvar_vector& obs, const dvar_vector& cv, const dvariable& wgt_dat)
  //pred=vector of predicted vals, obs=vector of observed vals, cv=vector of CVs in arithmetic space, wgt_dat=constant scaling of CVs
  //small_number is small value to avoid log(0) during search
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number=0.00001;
  dvar_vector var(cv.indexmin(),cv.indexmax()); //variance in log space
  var=log(1.0+square(cv/wgt_dat));   // convert cv in arithmetic space to variance in log space
  LkvalTmp=sum(0.5*elem_div(square(log(elem_div((pred+small_number),(obs+small_number)))),var) );
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;

//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
FUNCTION dvariable lk_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const double& ncomp, const double& minSS, const dvariable& wgt_dat)
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp = number of yrs in matrix, minSS=min N threshold, wgt_dat=scaling of N's
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number=0.00001;
  LkvalTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {if (nsamp(ii)>=minSS)
    {LkvalTmp-=wgt_dat*nsamp(ii)*sum(elem_prod((obs_comp(ii)+small_number),
               log(elem_div((pred_comp(ii)+small_number), (obs_comp(ii)+small_number)))));
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;


//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
FUNCTION dvariable lk_robust_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const double& ncomp, const dvariable& mbin, const double& minSS, const dvariable& wgt
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp = number of yrs in matrix, mbin=number of bins, minSS=min N threshold, wgt_dat=scaling of N's
  RETURN_ARRAYS_INCREMENT();
  dvariable LkvalTmp;
  dvariable small_number=0.00001;
  LkvalTmp=0.0;
  dvar_matrix Eprime=elem_prod((1.0-obs_comp), obs_comp)+0.1/mbin; //E' of Francis 2011, p.1131
  dvar_vector nsamp_wgt=nsamp*wgt_dat;
  //cout<<nsamp_wgt<<endl;
  for (int ii=1; ii<=ncomp; ii++)
  {if (nsamp(ii)>=minSS)
    {LkvalTmp+= sum(0.5*log(Eprime(ii))-log(small_number+mfexp(elem_div((-square(obs_comp(ii)-pred_comp(ii))) , (Eprime(ii)*2.0/nsamp_wgt(ii)) ))) );
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return LkvalTmp;


//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//Likelihood contribution: priors
FUNCTION  dvariable neg_log_prior(dvariable pred, const double& prior, dvariable var, int pdf)
  //prior=prior point estimate, var=variance (if negative, treated as CV in arithmetic space), pred=predicted value, pdf=prior type (1=none, 2=lognormal, 3=normal, 4=beta)
    dvariable LkvalTmp;
    dvariable alpha, beta, ab_iq;
    dvariable big_number=1e10;
    LkvalTmp=0.0;
    // compute generic pdf's
    switch(pdf) {
        case 1: //option to turn off prior
          LkvalTmp=0.0;
          break;
        case 2: // lognormal
          if(prior<=0.0) cout << "YIKES: Don't use a lognormal distn for a negative prior" << endl;
          else if(pred<=0) LkvalTmp=big_number=1e10;
          else {
            if(var<0.0) var=log(1.0+var*var) ;      // convert cv to variance on log scale
            LkvalTmp= 0.5*( square(log(pred/prior))/var + log(var) );
          }
        break;
        case 3: // normal
          if(var<0.0 && prior!=0.0) var=square(var*prior);       // convert cv to variance on observation scale
          else if(var<0.0 && prior==0.0) var=-var;               // cv not really appropriate if prior value equals zero
          LkvalTmp= 0.5*( square(pred-prior)/var + log(var) );
          break;
        case 4: // beta
          if(var<0.0) var=square(var*prior);          // convert cv to variance on observation scale
          if(prior<=0.0 || prior>=1.0) cout << "YIKES: Don't use a beta distn for a prior outside (0,1)" << endl;
          ab_iq=prior*(1.0-prior)/var - 1.0; alpha=prior*ab_iq; beta=(1.0-prior)*ab_iq;
          if(pred>=0 && pred<=1) LkvalTmp= (1.0-alpha)*log(pred)+(1.0-beta)*log(1.0-pred)-gammln(alpha+beta)+gammln(alpha)+gammln(beta);
          else LkvalTmp=big_number;
          break;
        default: // no such prior pdf currently available
          cout << "The prior must be either 1(lognormal), 2(normal), or 3(beta)." << endl;
          cout << "Presently it is " << pdf << endl;
          exit(0);
    }
    return LkvalTmp;

//--------------------------------------------------------------------------------
//SDNR: age comp likelihood (assumes fits are done with the robust multinomial function)
FUNCTION dvariable sdnr_multinomial(const double& ncomp, const dvar_vector& ages, const dvar_vector& nsamp,
                                const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const dvariable& wgt_dat)
  //ncomp=number of years of data, ages=vector of ages, nsamp=vector of N's,
  //pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, wgt_dat=likelihood weight for data source
```

```
  RETURN_ARRAYS_INCREMENT();
  dvariable SdnrTmp;
  dvar_vector o(1,ncomp);
  dvar_vector p(1,ncomp);
  dvar_vector ose(1,ncomp);
  dvar_vector res(1,ncomp);
  SdnrTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {
    o(ii)=sum(elem_prod(ages,obs_comp(ii)));
    p(ii)=sum(elem_prod(ages,pred_comp(ii)));
    ose(ii)=sqrt((sum(elem_prod(square(ages),pred_comp(ii)))-square(p(ii)))/(nsamp(ii)*wgt_dat));
  }
  res=elem_div((o-p),ose);
  SdnrTmp=sqrt(sum(square(res-(sum(res)/ncomp))/(ncomp-1.0)));
  RETURN_ARRAYS_DECREMENT();
  return SdnrTmp;


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

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

  if (last_phase())
  {
      // cout<<"start report"<<endl;
      get_weighted_current();
      //cout<<"got weighted"<<endl;
      get_msy();
      //cout<<"got msy"<<endl;
      get_miscellaneous_stuff();
      //cout<<"got misc stuff"<<endl;
      get_per_recruit_stuff();
      //cout<<"got per recruit"<<endl;
      get_effective_sample_sizes();
      get_Fmed_benchmarks();
      cout << "got Fmed benchmarks" << endl;

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

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


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

      sdnr_lc_gill=sdnr_multinomial(nyr_gill_lenc, lbins, nsamp_gill_lenc, pred_gill_lenc, obs_gill_lenc, w_lc_gill);

      sdnr_ac_cR=sdnr_multinomial(nyr_cR_agec, agebins, nsamp_cR_agec, pred_cR_agec, obs_cR_agec, w_ac_cR);

      sdnr_I_gill=sdnr_lognormal(pred_gill_cpue, obs_gill_cpue, gill_cpue_cv, w_I_gill);
      sdnr_I_seine=sdnr_lognormal(pred_seine_cpue, obs_seine_cpue, seine_cpue_cv, w_I_seine);


      //#########################################################################################
      //##  Passing parameters to vector for bounds check plotting
      //#########################################################################################
       Linf_out(8)=Linf; Linf_out(1,7)=set_Linf;
       K_out(8)=K; K_out(1,7)=set_K;
       t0_out(8)=t0; t0_out(1,7)=set_t0;
       len_cv_val_out(8)=len_cv_val; len_cv_val_out(1,7)=set_len_cv;
       log_R0_out(8)=log_R0; log_R0_out(1,7)=set_log_R0;
       steep_out(8)=steep; steep_out(1,7)=set_steep;
       rec_sigma_out(8)=rec_sigma; rec_sigma_out(1,7)=set_rec_sigma;
       R_autocorr_out(8)=R_autocorr; R_autocorr_out(1,7)=set_R_autocorr;
       selpar_L50_cR_out(8)=selpar_L50_cR; selpar_L50_cR_out(1,7)=set_selpar_L50_cR;
       selpar_slope_cR_out(8)=selpar_slope_cR; selpar_slope_cR_out(1,7)=set_selpar_slope_cR;
       selpar_L502_cR_out(8)=selpar_L502_cR; selpar_L502_cR_out(1,7)=set_selpar_L502_cR;
       selpar_slope2_cR_out(8)=selpar_slope2_cR; selpar_slope2_cR_out(1,7)=set_selpar_slope2_cR;
       selpar_age0_cR_out(8)=sel_age0_cR_logit; selpar_age0_cR_out(1,7)=set_sel_age0_cR;
```

```
      selpar_age1_cR_out(8)=sel_age1_cR_logit; selpar_age1_cR_out(1,7)=set_sel_age1_cR;
      selpar_age2_cR_out(8)=sel_age2_cR_logit; selpar_age2_cR_out(1,7)=set_sel_age2_cR;
      selpar_age3_cR_out(8)=sel_age3_cR_logit; selpar_age3_cR_out(1,7)=set_sel_age3_cR;
      selpar_age4_cR_out(8)=sel_age4_cR_logit; selpar_age4_cR_out(1,7)=set_sel_age4_cR;
      selpar_age0_cR2_out(8)=sel_age0_cR2_logit; selpar_age0_cR2_out(1,7)=set_sel_age0_cR2;
      selpar_age1_cR2_out(8)=sel_age1_cR2_logit; selpar_age1_cR2_out(1,7)=set_sel_age1_cR2;
      selpar_age2_cR2_out(8)=sel_age2_cR2_logit; selpar_age2_cR2_out(1,7)=set_sel_age2_cR2;
      selpar_age3_cR2_out(8)=sel_age3_cR2_logit; selpar_age3_cR2_out(1,7)=set_sel_age3_cR2;
      selpar_age4_cR2_out(8)=sel_age4_cR2_logit; selpar_age4_cR2_out(1,7)=set_sel_age4_cR2;
      selpar_L50_gill_out(8)=selpar_L50_gill; selpar_L50_gill_out(1,7)=set_selpar_L50_gill;
      selpar_slope_gill_out(8)=selpar_slope_gill; selpar_slope_gill_out(1,7)=set_selpar_slope_gill;
      log_q_gill_out(8)=log_q_gill; log_q_gill_out(1,7)=set_log_q_gill;
      log_q_seine_out(8)=log_q_seine; log_q_seine_out(1,7)=set_log_q_seine;
      M_constant_out(8)=M_constant; M_constant_out(1,7)=set_M_constant;
      log_avg_F_cR_out(8)=log_avg_F_cR; log_avg_F_cR_out(1,7)=set_log_avg_F_cR;

      log_rec_dev_output(styr_rec_dev,endyr_rec_dev)=log_rec_dev;
      log_F_dev_cR_out(styr_cR_L,endyr_cR_L)=log_F_dev_cR;
      M_dev_output(styr_seine_cpue,endyr_seine_cpue)=M_dev;
      log_Nage_dev_output(2,nages)=log_Nage_dev;

  #include "gm_make_Robject-001.cxx"   // write the S-compatible report

} //endl last phase loop
```

Appendix B Data inputs for the Beaufort Assessment Model

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  SEDAR 32A Gulf menhaden assessment August 2013
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: set-up section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#Starting and ending year of model
1977
2011

#Starting year to estimate recruitment deviation from S-R curve
1977

#Ending year to estimate recruitment deviation from S-R curve
2011

#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
1980
2008

#Ending year for first and second selectivity period
1993
2011

#Number of ages (5 classes is 0,...,4+) //assumes 4-6 is plus group
5
#Vector of agebins, last is a plus group
0.0 1.0 2.0 3.0 4.0

#Number length bins used to match length comps and number used to compute plus group
15
7

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
85 95 105 115 125 135 145 155 165 175 185 195 205 215      225
235 245 255 265 275 285 295


#Max value of F used in spr and msy calculations
10.0
#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


##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: observed data section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

######################Commercial Reduction Landings####################################################################
#Total landings (includes reduction landings + bait and recreational(very small; assumes a 100% discard mortality rate on rec landings))
#Starting and ending years of landings time series, respectively
1977
2011
#Observed Total removals (1000s of mt) and assumed CVs
447.60 820.60 779.83 702.50 553.71 855.53 925.26 985.12 884.39 824.02 906.06 634.45 582.22 538.64 549.67 429.43 548.67 771.77 472.02 491.75 623.15 493.65 692.49 580.29 522.11 575.07 517.70 469.24
434.13 464.65 454.09 425.57 457.69 379.93 613.83
0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04

#Number and vector of years of age compositions for commercial reduction fishery
35
1977  1978  1979  1980  1981  1982  1983  1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011
###sample sizes of age comps by year (first row observed Nsets, second row Nfish)
1492 1300 1163 1014 1042 1076 1485 1599 1324 1652 1647 1240 1392 1152 1164 1524 1537 1680 1470 1506 1124 1073 1183 969 740 836 1066 942 899 594 657 594 748 461 835
14897 12944 11121 9883 10273 10341 14523 15936 13225 16494 16458 12402 13950 11456 11378 14214 14576 16062 13489 12115 9923 9043 10641 8383 6222 5597 7839 6644 6206 4698 3989 4663 6193 3678 7254
#age composition samples (year,age) from recreational fishery--combined across gear and unweighted--last age is a plus group (1 to 12+)
0.000 0.763 0.218 0.018 0.001
0.000 0.708 0.286 0.005 0.001
0.000 0.593 0.363 0.043 0.001
0.009 0.472 0.452 0.060 0.007
0.000 0.763 0.189 0.044 0.005
0.000 0.571 0.366 0.056 0.007
0.000 0.526 0.428 0.043 0.003
0.000 0.697 0.259 0.039 0.004
0.000 0.758 0.218 0.020 0.003
0.000 0.456 0.522 0.019 0.003
0.000 0.603 0.358 0.038 0.001
0.000 0.660 0.319 0.019 0.002
0.000 0.766 0.224 0.009 0.000
0.000 0.668 0.306 0.023 0.002
0.000 0.462 0.487 0.045 0.006
```

```
0.000 0.559 0.384 0.050 0.007
0.001 0.666 0.292 0.037 0.004
0.000 0.496 0.437 0.060 0.007
0.000 0.351 0.622 0.026 0.001
0.000 0.391 0.550 0.055 0.004
0.000 0.544 0.403 0.046 0.007
0.000 0.392 0.563 0.041 0.004
0.000 0.544 0.386 0.067 0.003
0.000 0.362 0.564 0.062 0.012
0.000 0.250 0.672 0.074 0.005
0.000 0.317 0.573 0.107 0.003
0.000 0.362 0.571 0.064 0.003
0.000 0.560 0.353 0.080 0.008
0.019 0.394 0.541 0.043 0.003
0.000 0.459 0.470 0.065 0.006
0.000 0.463 0.510 0.024 0.004
0.000 0.266 0.683 0.044 0.006
0.000 0.126 0.731 0.129 0.013
0.000 0.529 0.404 0.061 0.006
0.007 0.632 0.317 0.037 0.007


#######################################LA gill net index############################################################
#Starting and ending years of LA gill net index
1988
2011
#Observed index and assumed CVs
0.27 0.21 0.24 0.25 0.22 0.41 0.84 0.55 0.67 1.24 0.94 0.80 1.00 1.34 0.99 1.02 0.90 1.24 1.18 0.86 3.52 2.64 0.75 1.93
0.09 0.10 0.10 0.11 0.13 0.16 0.15 0.15 0.14 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.14 0.13 0.12 0.13 0.12 0.11 0.14 0.13

#Number and vector of years of length compositions for gill net survey
16
1996  1997  1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011
#sample size of gill net survey length comp data by year (first row observed Ntrips, second row Nfish)
209 236 246 231 266 219 241 247 231 234 297 253 277 308 193 248
2338 3386 3059 2913 4235 3542 3131 3297 2851 3659 4588 3301 5794 5782 2377 4222
#gill net length composition samples (year,lengthbin 1 cm)
0.01 0.01 0.01 0.02 0.07 0.15 0.16 0.09 0.11 0.11 0.08 0.05 0.05 0.04 0.03
0.00 0.01 0.00 0.01 0.06 0.13 0.14 0.11 0.1 0.13 0.09 0.07 0.07 0.03 0.03
0.00 0.01 0.01 0.02 0.06 0.17 0.19 0.11 0.11 0.12 0.07 0.04 0.03 0.02 0.02
0.00 0.01 0.01 0.03 0.07 0.16 0.14 0.1 0.11 0.11 0.08 0.06 0.06 0.03 0.02
0.00 0.00 0.01 0.02 0.06 0.14 0.11 0.07 0.07 0.12 0.12 0.08 0.08 0.06 0.04
0.01 0.01 0.01 0.01 0.06 0.13 0.11 0.08 0.13 0.14 0.08 0.06 0.07 0.05 0.06
0.00 0.01 0.01 0.03 0.06 0.14 0.14 0.08 0.09 0.11 0.09 0.06 0.07 0.06 0.05
0.00 0.01 0.01 0.03 0.09 0.19 0.19 0.1 0.1 0.13 0.07 0.02 0.02 0.02 0.01
0.00 0.01 0.02 0.04 0.11 0.18 0.17 0.11 0.1 0.08 0.07 0.04 0.04 0.02 0.01
0.00 0.01 0.01 0.03 0.06 0.14 0.18 0.12 0.13 0.13 0.08 0.04 0.04 0.02 0.00
0.01 0.01 0.01 0.02 0.08 0.16 0.16 0.15 0.1 0.1 0.08 0.05 0.04 0.02 0.01
0.01 0.01 0.01 0.02 0.04 0.13 0.17 0.15 0.14 0.13 0.07 0.04 0.03 0.02 0.01
0.00 0.01 0.01 0.02 0.04 0.12 0.14 0.1 0.12 0.15 0.11 0.08 0.06 0.03 0.02
0.00 0.01 0.01 0.02 0.04 0.08 0.13 0.12 0.11 0.13 0.12 0.07 0.07 0.05 0.03
0.00 0.01 0.01 0.03 0.07 0.14 0.12 0.09 0.09 0.11 0.1 0.08 0.06 0.05 0.03
0.00 0.01 0.02 0.03 0.09 0.17 0.16 0.09 0.07 0.11 0.1 0.06 0.05 0.02 0.01


#######################################Seine survey index############################################################
#Starting and ending years of the seine survey index
1996
2010
#Observed CPUE and assumed CVs
1.22 0.44 0.98 0.77 0.33 0.67 0.89 1.09 0.72 1.50 0.62 0.76 0.46 1.56 2.97
0.20 0.19 0.20 0.20 0.23 0.20 0.19 0.18 0.20 0.21 0.21 0.19 0.21 0.20 0.20



##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: parameter section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##################Parameter values and initial guesses#########################################################################################
#####################################################
##prior PDF (1=none, 2=lognormal, 3=normal, 4=beta)
#############################################################
#initial # lower # upper #       #  prior  # prior   # prior #
# guess  # bound # bound # phase #  mean   # var/-CV #  PDF  #
#--------#-------#-------#-------#---------#---------#-------#
                                                    ###### Biological input ###################################
 239.5      150     400      3     239.5    0.087      1     # VonBert Linf (units in mm FL)
 0.40       0.05    0.8      3     0.40     -0.27      1     # VonBert K (units in mm FL)
 -0.78    -4.0     0.0      3     -1.013   -0.28      1     # VonBert t0 (units in mm FL)
 0.2        0.01    0.5        3     0.08     0.235      1     # CV of length at age
 0.26       0.05    0.65     -3     0.26     0.092      1     # constant M (used only to compute MSST=(1-M)SSBmsy)
                                                     ###### SR parameters ######################################
 0.75       0.21    0.99     -4     0.80     0.0196     1     # SR steepness parameter
 5.3       2.0     10.0      1     5.3     -0.25       1     # SR log_R0 parameter
 0.0       -1.0     1.0      -3     0.0      -0.5       1     # SR recruitment autocorrelation (lag 1)
 0.6        0.2     1.2      -4     0.6      -0.25      1     # s.d. of recruitment in log space
                                                     ###### Selectivity parameters ##############################
 1.25       0.75     10.0     -1     1.25     -0.5       1     # reduction age at 50% selectivity
 4.08       0.5     12.0      -1     4.08     -0.5       1     # reduction slope of ascending limb
 3.75       0.5     10.0      -3     2.75     -0.5       1     # reduction age at 50% selectivity for descending limb
 4.08       0.5     12.0      -3     4.08     -0.5       1     # reduction slope of descending limb

 -9.0    -10.0     10.0      -1     -5.0     -0.5      1     #age-0 cR selectivity in logit space
 0.0      -10.0     10.0       1      0.0     -0.5      1     #age-1 cR selectivity in logit space
 9.0     -10.0     10.0      -1      5.0     -0.5      1     #age-2 cR selectivity in logit space
 -0.6      -15.0     10.0       -1      0.0     -0.5      1     #age-3 cR selectivity in logit space
 0.0      -10.0     10.0       -1      0.0     -0.5      1     #age-4+ cR selectivity in logit space

 -9.0    -10.0     10.0      -1     -5.0     -0.5      1     #age-0 cR selectivity in logit space-period 2
 0.0      -10.0     10.0       -1      0.0     -0.5      1     #age-1 cR selectivity in logit space-period 2
 9.0     -10.0     10.0      -1      5.0     -0.5      1     #age-2 cR selectivity in logit space-period 2
 0.0      -15.0     10.0      -1      0.0     -0.5      1     #age-3 cR selectivity in logit space-period 2
 0.0      -10.0     10.0      -1      0.0     -0.5      1     #age-4+ cR selectivity in logit space-period 2
```

```
 1.25       0.25     10.0      2     1.25     -0.5      1     # gill net survey age at 50% selectivity
 4.08       2.5     25.0      2     4.08     -0.5       1     # gill net survey slope of ascending limb
                                                              ###### Index catchability parameters ########################
 -10.0      -15   10.0        1     -10.0    -0.5      1     # gill net index (log q)
 -10.0      -15    0.0        1     -10.0    -0.5      1     # seine index (log q)
                                                              ###### Fishing mortality parameters ########################
 -1.0       -10.0   4.50        1     -1.0     -0.5      1     #cR average log mean F

##### Dev vectors ###################################################################################
#########################
# lower # upper #       #
# bound # bound # phase #
#-------#-------#-------#
   -5      5       2    # cR F devs
   -5      5      3    # rec devs
   -1.25    2.00     -2    # M devs for age-1
   -30     30        2     #devs for initial N or age structure


# cR F dev initial guesses
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#-1.67222403071 -1.59880268257 -1.53003732453 -1.46484809148 -1.40160106482 -1.34187427116 -1.28539715100 -1.22891896632 -1.17527883036 -1.12406892691 -1.07479213292 -1.00764268639 -0.943226356126 -0.88122529863

# rec devs
0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0       0
#0.194201111886 -1.13978478939 -0.0661388106143 0.165700304946 1.22456944710 -0.444668148641 0.265368070225 1.18726157682 -0.652591080236 -0.705755253699 0.812035361192 -0.612963572894 0.479884734830 -1.31433184

# M devs for age-1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

#N devs
0 0 0 0



##-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;
##-- BAM DATA SECTION: likelihood weights section
##-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;

##################Likelihood Component Weighting#################################################################################
##Weights in objective fcn  (commented wgts are those after correcting M vector--did not use)
1.0         #landings
0.132#0.161#0.164#1.0         #gill net length comps
0.021#0.110#0.021#1.0         #commercial age comps
0.615#0.375#0.362#0.354#1.0         #gill net index
0.429#0.508#0.528#1.0         #seine index
0#2.0         #M_devs
1.0     #S-R residuals
0.0     #constraint on early recruitment deviations
0.0     #constraint on ending recruitment deviations
0.0         #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization) fULL F summed over fisheries
0.0     #weight on tuning F (penalty not applied in final phase of optimization)



##-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;
##-- BAM DATA SECTION: miscellaneous stuff section
##-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;-->&lt;&gt;

#length-weight (FL-whole wgt) coefficients a and b, W=aL^b, (W in g, FL in mm)--sexes combined
8.17E-6
3.18

#time-invariant vector of % maturity-at-age for females (ages 0-4+)
0.0  0.0  1.00  1.00  1.00

#time-invariant vector of proportion female (ages 0-4+)--assume 50:50 sex ratio
0.5 0.5 0.5 0.5 0.5

#time-invariant fecundity at age (number of maturing ova per individual)
#0.0    8728 21814 36385 49542
0.0 9153 21234 35341 49118


#time-invariant weight (in grams) at age at spawning
#0.0    45.3 95.9 145.9 187.9
0.0     47.1 93.9 142.5 186.6

#time-invariant weight (in grams) at age at start of fishing year
34.2 69.9 121.6 168.1 205.3

#time of year (as fraction) for spawning: Jan 1 = 0d/365d
0.0

#age-dependent natural mortality at age (ages 0-4+)
1.62 1.30 1.10 1.00 0.94
#Max observed age
7

#Spawner-recruit parameters
# SR function switch (integer 1=Beverton-Holt, 2=Ricker)
1

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

```
2
#Random walk switch:Integer value (choose estimation phase, negative value turns it off)
-3
#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence
0.03
0.03
0.03

#Tuning F (not applied in last phase of optimization)
0.2
#Year for tuning F
2011

##threshold sample sizes for length comps (set to 99999.0 if sel is fixed)
1.0 #gill net

#threshold sample sizes (greater than or equal to) for age comps
1.0 #commerical reduction

#Ageing error matrix (columns are true age 0-4+, rows are ages as read for age comps: columns should sum to one)
1.00 0.00 0.00 0.00 0.00
0.00 1.00 0.11 0.00 0.00
0.00 0.00 0.78 0.16 0.00
0.00 0.00 0.11 0.68 0.17
0.00 0.00 0.00 0.16 0.83

999 #end of data file flag
```