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 a statistical catch-age model (Quinn and Deriso 1999), implemented with the AD Model Builder software (ADMB Project 2009). In essence, a statistical catch-age model simulates a population forward in time while including fishing processes. Quantities to be estimated are systematically varied until characteristics of the simulated populations match available data on the real population. Statistical catch-age models share many attributes with ADAPT-style tuned and untuned VPAs.

The method of forward projection has a long history in fishery models. It was introduced by Pella and Tomlinson (1969) for fitting production models and then used by Fournier and Archibald (1982), Deriso et al. (1985) in their CAGEAN model, and Methot (1989) in his Stock Synthesis model. The catch-age model of this assessment is similar in structure to the CAGEAN and stock-synthesis models. Previous versions of this assessment model have been used in SEDAR assessments of reef fishes in the U.S. South Atlantic, such as red porgy, black sea bass, tilefish, snowy grouper, gag grouper, greater amberjack, vermilion snapper, Spanish mackerel, red grouper, red snapper, as well as for assessments of Atlantic and gulf menhaden. The present version of this model, customized for SEDAR 27 gulf menhaden, is described below.

2 Model configuration and equations

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

Stock dynamics In the assessment model, new biomass was acquired through growth and recruitment, while abundance of existing cohorts experienced exponential decay from fishing and natural mortality. The population was assumed closed to immigration and emigration. The model included age classes $0 - 4^+$, where the oldest age class 4^+ allowed for the accumulation of fish (i.e., plus group). Initial numbers at age were estimated in the model. but were penalized if they deviated from the stable age structure that resulted by assuming a constant, historical fishing mortality equal to the geometric mean fishing mortality for the first three years following model implementation (i.e., the geometric mean fishing mortality from 1948-1950).

Natural mortality rate The natural mortality rate (M) was assumed constant over time, but decreasing with age. The form of M as a function of age was based on Lorenzen (1996). The Lorenzen (1996) approach inversely relates the natural mortality at age to mean weight at age W_a by the power function $\mathrm{M}_a{=}\alpha W_a^\beta$, where α is a scale parameter and β is a shape parameter. Lorenzen (1996) provided point estimates of $\alpha = 3.69$ and $\beta = -0.305$ for oceanic fishes, which were used for this assessment. The Lorenzen estimates of M_a were rescaled so that age-2 natural mortality was the value of 1.10 estimated during a tagging study (Ahrenholz 1981).

Growth Mean size at age of the population (fork length, FL) was modeled with the von Bertalanffy equation with parameters estimated externally using all years of data, and annual weight at age was a model input. For fitting length composition data, the distribution of size at age was assumed normal with a cv for each age, which was fixed for each age within the assessment model and was estimated externally based on all years of available data.

Maturity Maturity at age was a constant vector over time provided by the DW.

Spawning biomass Spawning biomass was modeled as fecundity of the population at the time of spawning, where sex ratio at age (50:50) was provided by the DW. For gulf menhaden, peak spawning was considered to occur on January 1.

Recruitment Recruitment was predicted from spawning biomass using a Beverton–Holt spawner-recruit model. Steepness, h, was a key parameter of this model and was estimated. Recruitment deviations were estimated starting in 1948.

Landings Time series of landings from three fisheries were combined: commercial reduction, bait, and recreational. Bait and recreational landings were very small compared to commercial reduction landings, thus all landings were aggregated. Landings were modeled with the Baranov catch equation (Baranov 1918) and were fitted in 1000s of metric tons.

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

Selectivity at age applied to landings was estimated for each year; thus age and time varying selectivity was used. Selectivity applied to the landings was broken into three time periods. The first was 1948-1963, which was assumed to be the average selectivity from the years 1964-1966, because age composition data were not available before 1964. Selectivity for the period 1964 to 1979 was assumed zero for age-0, one for age-2, and was estimated for ages 1, 3, and 4. A diffuse prior was used for estimating selectivity for each age in this time period to help with optimization. Priors help by steering estimation away from parameter space with no response in the likelihood surface. Selectivity for the period 1980 to 2010 was assumed to be flat-topped, and thus assumed a selectivity of zero for age-0, one for ages-2+, and was estimated for age-1. No priors were used in the most recent time period.

Discards Discards of gulf menhaden were assumed to be miniscule and therefore were not modeled. However, a sensitivity run was completed that included discards from shrimp trawls.

Indices of abundance The model was fit to two fishery independent indices of abundance (gillnet index 1986–2010; seine juvenile abundance index 1977–2010). Predicted indices were conditional on selectivity of the survey/gear and were computed from numbers at age at the midpoint of the year for the gillnet index and, in the case of seine index, as numbers of age zero individuals. Catchability was assumed constant for the gulf menhaden fishery independent indices.

Selectivity at age applied to the gillnet index was estimated. One selectivity was estimated for each age for the entire time period of the gillnet index (1986-2010). Length composition data were used for estimating the selectivity parameters. A diffuse prior was used for estimating selectivity for each age. The prior was only used to provide weak information to help the optimization routine during model execution.

Selectivity of the juvenile abundance index based on seine surveys was assumed to be one at age-0 and zero at all other ages.

Biological reference points Biological reference points (benchmarks) were calculated based on maximum sustainable yield (MSY) estimates from the Beverton–Holt spawner-recruit model with bias correction. Computed benchmarks included MSY, fishing mortality rate at MSY (F_{MSY}), and spawning biomass (total fecundity) at MSY (SSB_{MSY}). These benchmarks are conditional on the estimated selectivity functions. The selectivity pattern used here was the effort-weighted selectivities at age, with effort from the fishery estimated as the full F averaged (geometric) over the last three years of the assessment.

Fitting criterion The fitting criterion was a penalized likelihood where model predictions of landings, composition data, and abundance indices were compared with available data using lognormal (landings and indices) and multinomial (length and age composition) likelihood functions.

The model included the capability for each component of the likelihood to be weighted by user-supplied values (for instance, to give more influence to desired data sources). However, for initial runs of the gulf menhaden assessment model, all weights were set to 1.0 for the data components. Iterative reweighting was then used to change the weights based on the standard deviation of the normalized residuals (Francis 2011). Then, the weights were changed based on improving the fits to the indices (Francis 2011).

In addition to likelihoods, the capability of several penalties and prior distributions were included in the compound objective function. Priors and penalties were applied to maintain parameter estimates near reasonable values, and to prevent the optimization routine from drifting into parameter space with negligible gradient in the likelihood.

Model testing Experiments with a reduced model structure indicated that parameters estimated from the BAM were unbiased and could be recovered from simulated data with little noise (SEDAR 2007). Further, the general model structure has been through multiple SEDAR reviews. As an additional measure of quality control, 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

ADMB Project, 2009. AD Model Builder: automatic differentiation model builder. Available: http://www.admb-project.org.

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

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

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

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

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

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

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

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

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

SEDAR, 2009a. SEDAR 19 Data Workshop Report.

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

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

Table 2.1. General definitions, input data, population model, and negative log-likelihood components of the statistical catch-age model applied to 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	$y \in \{1948 \ldots 2010\}$
Index of ages	a	$a \in \{0 \ldots A\}$, where $A = 4^+$
Index of selectivity periods	r	$r \in \{1 \ldots 3\}$ where 1 = commercial reduction fishery 1948-1979, 2 = commercial reduction fishery 1980-2010, 3 = gillnet index
Length bins	l	$l \in \{5, 15, \ldots, 505\text{mm}\}$, with midpoint of 10mm bin used to match length compositions. Largest 10 length bins treated as a plus group.
Index of fisheries	f	f represents the commercial reduction fishery, and a small amount of landings from the bait and recreational fisheries.
Index of CPUE	u	$u \in \{1 \ldots 2\}$ where 1 = gillnet index, 2 = seine juvenile abundance index
Input Data		
Proportion female at age	ρ_a	Considered constant (50:50) across years and ages
Proportion mature at age	m_a	Proportion mature is zero at ages zero and one and is one at ages two plus; assumed constant across years
Annual fecundity at age	$\mathcal{F}_{y,a}$	where fecundity was a model input from the DW based on Lewis and Roithmayr (1981)
Observed length compositions	$p^{\lambda}_{u,l,y}$	Proportional contribution of length bin l in year y to index u
Observed age compositions	$p^{\alpha}_{a,y}$	Proportional contribution of age class a in year y to the commercial reduction fishery.
Ageing error matrix	$\mathcal{E}$	Estimated from ageing scales paired with otoliths.
Length composition sample sizes	$n^{\lambda}_{u,y}$	Effective number of length samples collected in year y from index u
Age composition sample sizes	n^{α}_{y}	Effective number of age samples collected in year y from the commercial reduction fishery.
Observed landings	L_y	Reported landings in year y from the commercial reduction fishery and small bait and recreational fisheries. Landings, L, in 1000s of metric tons.
CVs of landings	c^{L}_{y}	Assumed 0.04 in arithmetic space.
Observed abundance indices	$U_{u,y}$	$u = 1$, gillnet index (numbers), $y \in \{1986 \ldots 2010\}$ $u = 2$, seine juvenile abundance index (numbers), $y \in \{1977 \ldots 2010\}$
CVs of abundance indices	$c^{U}_{u,y}$	$u = \{1 \ldots 2\}$ as above. Annual values estimated from delta-lognormal GLM with jackknifing. Each time series was scaled to its mean
Natural mortality rate	M_a	Function of weight at age (w_a): $M_a = \alpha w_a^{\beta}$, with estimates of α and β from Lorenzen (1996). Lorenzen M_a then rescaled to tagging estimates of natural mortality (Ahrenholz 1981).

Table 2.1. (continued)

Quantity	Symbol	Description or definition
Population Model		
Predicted mean length at age	$\widehat{l_a}$	Fork length (midyear); $\widehat{l_a} = L_\infty(1 - \exp[-K(a - t_0 + 0.5)])$ where K, L_∞, and t_0 were fixed parameters within the assessment model.
CV of l_a	$\widehat{c}_a^\lambda$	Variation of growth for each age was fixed and assumed constant across years.
Age–length conversion of population	$\psi^u_{a,l}$	$\psi^u_{a,l} = \frac{1}{\sqrt{2\pi}(\widehat{c}_a^\lambda \widehat{l_a})} \exp \frac{\left[-\left(l_l - \widehat{l_a}\right)^2\right]}{\left(2\left(\widehat{c}_a^\lambda \widehat{l_a}\right)^2\right)}$, the Gaussian density function. Matrix ψ^u is rescaled to sum to one within ages, with the largest age a plus group. This matrix is constant across years and is used only to match length comps of fishery independent indices.
Individual weight at age and weight at age at time of spawning	$w_{a,y}$	Computed from annual length at age by $w_{a,y} = \theta_{1,y} l_{a,y}^{\theta_{2,y}}$ where $\theta_{1,y}$ and $\theta_{2,y}$ are parameters from the DW. Weight at age at the beginning of the year, or during spawning, which represents January 1, was estimated using annual Von Bertalanffy growth equations and the weight-length equation.
Fishery and index selectivities	$\widehat{s}_{(f,u),a,r}$	$\widehat{s}_{(f,u),a,r}$ is an estimated parameter for each age modeled on the logit scale.
Fishing mortality rate of landings	$F_{a,y}$	$F_{a,y} = \widehat{s}_{f,a,y} \widehat{F}_y$ where $\widehat{F}_y$ is an estimated fully selected fishing mortality rate and $s_{f,a,y} = s_{f,a,r}$ for y in the years represented by r.
Total mortality rate	$Z_{a,y}$	$Z_{a,y} = M_a + F_{a,y}$
Apical F	F_y	$F_y = \max(F_{a,y})$
Abundance at age at time of spawning	$N_{a,y}$	$N_{0,1948} = \frac{\widehat{R}_0(0.8\varsigma\widehat{h}S_{equil} - 0.2\phi_0(1-\widehat{h}))}{(\widehat{h}-0.2)S_{equil}}$ $\widehat{N}_{1+,1948}$ estimated subject to penalties for deviating from equilibrium conditions expected given assumptions about initial fishing mortality (see "Objective Function") $N_{0,y+1} = \frac{0.8\widehat{R}_0\widehat{h}S_{y+1}}{0.2\phi_0\widehat{R}_0(1-\widehat{h})+(\widehat{h}-0.2)S_{y+1}} \exp(\widehat{R}_{y+1})\text{for} \quad y \geq 1948$ $N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (1 \ldots A-1)$ $N_{A,y} = N_{A-1,y-1}\frac{\exp(-Z_{A-1,y-1})}{1-\exp(-Z_{A,y-1})}$ Parameters $\widehat{R}_0$ (asymptotic maximum recruitment) and $\widehat{h}$ (steepness) are estimated parameters of the spawner-recruit curve, component of the spawner-recruit curve, and $\widehat{R}_y$ are estimated annual recruitment deviations in log space. The bias correction is $\varsigma = \exp(\widehat{\sigma}^2/2)$, where $\widehat{\sigma}^2$ is the variance of recruitment deviations. Quantities ϕ_0, S_y, and S_{equil} are described below.
Abundance at age (mid-year)	$N'_{a,y}$	Used to match indices of abundance $N'_{a,y} = N_{a,y}\exp(-Z_{a,y}/2)$
Unfished abundance at age per recruit at time of spawning	NPR_a	$NPR_1 = 1 \times \exp(-t_{\text{spawn}} M_1)$ $NPR_{a+1} = NPR_a \exp[-(M_a(1-t_{\text{spawn}}) + M_{a+1}t_{\text{spawn}})] \quad \forall a \in (1 \ldots A-1)$ $NPR_A = \frac{NPR_{A-1}\exp[-(M_{A-1}(1-t_{\text{spawn}})+M_A t_{\text{spawn}})]}{1-\exp(-M_A)}$
Unfished spawning biomass per recruit	$\phi_{0,y}$	$\phi_{0,y} = \sum_{a=1}^{A} NPR_a \rho_a m_a \mathcal{F}_{y,a}$ In units of fecundity. This is also computed overall using a constant vector of fecundity, which is the average fecundity over the last few years.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Spawning biomass	S_y	$\sum_{a=1}^{A} N''_{a,y}\rho_a m_a \mathcal{F}_a$ In units of fecundity.
Initialization mortality at age	Z_a^{init}	$Z_a^{init} = M_a + s_a^{init}\widehat{F}^{init}$ where $\widehat{F}^{init}$ is an estimated initialization F, and s_a^{init} is the initialization selectivity.
Initial equilibrium abundance at age	N_a^{equil}	Equilibrium age structure given Z_a^{init}
Initial equilibrium spawning biomass	S_{equil}	$S_{equil} = \sum_{a=1}^{A} N_a^{equil}\rho_a m_a \mathcal{F}_{1,a}$
Population biomass	B_y	$B_y = \sum_a N_{a,y} w_a$
Landings at age in numbers	$L'_{a,y}$	$L'_{a,y} = \frac{F_{a,y}}{Z_{a,y}} N_{a,y}[1 - \exp(-Z_{a,y})]$
Landings at age in weight	$L''_{a,y}$	$L''_{a,y} = w_{a,y} L'_{a,y}$
Index catchability	$\widehat{q}_u$	Catchability was estimated as a constant for each index u.
Predicted landings	$\breve{L}_y$	$\breve{L}_y = \sum_a L''_{a,y}$
Predicted length compositions of fishery independent data	$\breve{p}^{\lambda}_{u,l,y}$	$\breve{p}^{\lambda}_{u,l,y} = \frac{\sum_a \psi^u_{a,l} s_{u,a,y} N'_{a,y}}{\sum_a s_{u,a,y} N'_{a,y}}$
Predicted age compositions	$\breve{p}^{\alpha}_{a,y}$	$\breve{p}^{\alpha}_{a,y} = \frac{\mathcal{E} L'_{a,y}}{\sum_a L'_{a,y}}$ this formulation accounts for ageing error.
Predicted CPUE	$\breve{U}_{u,y}$	$\breve{U}_{u,y} = \begin{cases} \widehat{q}_u \sum_a N'_{a,y} s_{u,a} & : \quad u = 1 \\ \widehat{q}_u N'_{a=0,y} & : \quad u = 2 \end{cases}$ where $s_{u,a}$ is the selectivity of the relevant fishery independent survey in the year corresponding to y.
Objective Function		
Multinomial length compositions	Λ_1	$\Lambda_1 = -\omega_1 \sum_u \sum_y \left[n^{\lambda}_{u,y} \sum_l (p^{\lambda}_{u,l,y} + x) \log \left(\frac{(\breve{p}^{\lambda}_{u,l,y} + x)}{(p^{\lambda}_{u,l,y} + x)} \right) \right]$ where ω_1 is a preset weight and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term. Bins are 10 mm wide.
Multinomial age compositions	Λ_2	$\Lambda_2 = -\omega_2 \sum_y \left[n^{\alpha}_y \sum_a (p^{\alpha}_{a,y} + x) \log \left(\frac{(\breve{p}^{\alpha}_{a,y} + x)}{(p^{\alpha}_{a,y} + x)} \right) \right]$ where ω_2 is a preset weight and x =1e-5 is an arbitrary value to avoid log zero. The denominator of the log is a scaling term.
Lognormal landings	Λ_3	$\Lambda_3 = \omega_3 \sum_y \frac{\left[\log\left((L_y + x) \big/ (\breve{L}_y + x) \right) \right]^2}{2(\sigma^L)^2}$ where ω_3 is a preset weight and x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, $\sigma^L = \sqrt{\log(1 + (c^L_y)^2)}$.

Table 2.1. (continued)

Quantity	**Symbol**	**Description or definition**
Lognormal CPUE	Λ_4	$\Lambda_4 = \sum\limits_u \omega_4^u \sum\limits_y \frac{\left[\log\left((U_{u,y}+x)\big/(\breve{U}_{u,y}+x)\right)\right]^2}{2(\sigma_u^U)^2}$ where ω_4^u is a preset weight and x =1e-5 is an arbitrary value to avoid log zero or division by zero. Here, Here, $\sigma_u^U = \sqrt{\log(1 + (c_{u,y}^U)^2)}$.
Lognormal recruitment deviations	Λ_5	$\Lambda_5 = \omega_5 \left[R_{1948}^2 + \sum\limits_{y>1948} \frac{[(R_y - \widehat{\varrho} R_{y-1}) + (\widehat{\sigma}_R^2 \big/ 2)]^2}{2\widehat{\sigma}_R^2} \right]$ where R_y are recruitment deviations in log space, $\omega_5 = 1$ is a preset weight, $\widehat{\varrho}$ is the estimated first-order autocorrelation (not used for the SEDAR 27 gulf menhaden base run), and $\widehat{\sigma}_R^2$ is the recruitment variance.
Penalty on initial age structure	Λ_6	$\Lambda_6 = \sum\limits_{a=2}^{A} (\widehat{N}_{a,1948} - N_a^{equil})^2$
Prior distributions and penalties	Λ_7	Several prior distributions were imposed on parameters to keep them in reasonable parameter spaces. This included priors on gillnet selectivity for ages 1, 3, and 4 and on commercial reduction fishery selectivity from 1964 to 1979 on ages 1, 3, and 4 of the form: $\Lambda_7 = 0.5 * (\frac{(pred_{(f,u),a} - prior_{(f,u),a})^2}{prior_{(f,u),a}} + log(prior_{(f,u),a}))$ where expected values priors were supplied as input.
Total objective function	Λ	$\Lambda = \sum\limits_{i=1}^{7} \Lambda_i$ Objective function minimized by the assessment model

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

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  GSMFC Assessment: Gulf Menhaden, July 2011
//##
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

DATA_SECTION

!!cout << "Starting Gulf Menhaden Assessment Model" << endl;

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

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

//3 periods of size regs: styr-83 no restrictions, 1984-91 12-inch TL, 1992-08 20-in TL
init_int endyr_period1;
init_int endyr_period2;
init_int endyr_period3;

//end of first period of selectivity for gillnet survey and length comps
init_int endyr_period1_gill;

//starting and ending years to use for benchmark calculations
init_int styr_bench;
init_int endyr_bench;

//Total number of ages
init_int nages;

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

//Number of length bins and number in plus group
init_int nlen;
init_int nlenplus;
number nlenbins;
 LOCAL_CALCS
   nlenbins=nlen-nlenplus;
 END_CALCS

//Vectors of length bins and vector of plus group length bins
init_ivector lenbinstotal(1,nlen);
init_ivector lenplusbins(1,nlenplus);
init_ivector lenbins(1,nlenbins);

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

//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;
//Female maturity and proportion female at age
init_vector maturity_f_obs(1,nages);            //proportion females mature at age
init_vector maturity_m_obs(1,nages);            //proportion males mature at age
init_vector prop_f_obs(1,nages);                //proportion female at age
init_number spawn_time_frac; //time of year of peak spawning, as a fraction of the year
// Natural mortality
init_vector set_M(1,nages); //age-dependent: used in model
init_number set_M_constant; //age-independent: used only for MSST
init_matrix set_M_mat(styr,endyr,1,nages); //age and year specific M
//Spawner-recruit parameters (Initial guesses or fixed values)
init_number set_SR_switch;  //switch for SR curve
init_number set_steep;      //recruitment steepness
init_number set_steep_se;   //SE or recruitment steepness
init_number set_log_R0;     //Recruitment R0
init_number set_R_autocorr; //Recruitment autocorrelation
init_number set_rec_sigma;  //recruitment standard deviation in log space
init_number set_rec_sigma_se;//SE of recruiment standard deviation in log space

//--><>--><>--><>-- Weight-at-age in the fishery (g) --><>--><>--><>--><>--><>--><>--><>--><>
init_matrix wgt_fish_g(styr,endyr,1,nages);
```

```
//--><>--><>--><>-- Weight-at-age for the spawning population - start of year (g) --><>--><>--><>--><>
init_matrix wgt_spawn_g(styr,endyr,1,nages);

//--><>--><>--><>-- Fecundity-at-age - not adjusted for maturity (trillions) --><>--><>--><>--><>
init_matrix fec_eggs(styr,endyr,1,nages);

//--><>--><>--><>-- Juvenile Abundance Index from seine surveys --><>--><>--><>--><>--><>
init_int JAI_cpue_switch;
//CPUE
init_int styr_JAIs_cpue;
init_int endyr_JAIs_cpue;
init_vector obs_JAIs_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);   //Observed CPUE
init_vector JAIs_cpue_cv(styr_JAIs_cpue,endyr_JAIs_cpue);    //CV of cpue

//--><>--><>--><>-- Juvenile Abundance Indices from seine surveys --><>--><>--><>--><>--><>
//CPUE, must have zeros in place of missing values
init_vector obs_JAI1_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);   //Observed CPUE 1
init_vector JAI1_cpue_cv(styr_JAIs_cpue,endyr_JAIs_cpue);    //CV of cpue 1
init_vector obs_JAI2_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);   //Observed CPUE 2
init_vector JAI2_cpue_cv(styr_JAIs_cpue,endyr_JAIs_cpue);    //CV of cpue 2
init_vector obs_JAI3_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);   //Observed CPUE 3
init_vector JAI3_cpue_cv(styr_JAIs_cpue,endyr_JAIs_cpue);    //CV of cpue 3
init_vector obs_JAI4_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);   //Observed CPUE 4
init_vector JAI4_cpue_cv(styr_JAIs_cpue,endyr_JAIs_cpue);    //CV of cpue 4

//--><>--><>--><>-- Juvenile Abundance Index from trawl surveys --><>--><>--><>--><>--><>
//CPUE
init_int styr_JAIt_cpue;
init_int endyr_JAIt_cpue;
init_vector obs_JAIt_cpue(styr_JAIt_cpue,endyr_JAIt_cpue);   //Observed CPUE
init_vector JAIt_cpue_cv(styr_JAIt_cpue,endyr_JAIt_cpue);    //CV of cpue

//--><>--><>--><>-- Adult abundance index from gillnet surveys --><>--><>--><>--><>--><>
//CPUE
init_int styr_gill_cpue;
init_int endyr_gill_cpue;
init_vector obs_gill_cpue(styr_gill_cpue,endyr_gill_cpue);   //Observed CPUE
init_vector gill_cpue_cv(styr_gill_cpue,endyr_gill_cpue);    //cv of cpue

// Length Compositions (10 mm bins)
init_int nyr_gill_lenc;
init_int styr_gill_lenc;
init_int endyr_gill_lenc;
init_ivector yrs_gill_lenc(1,nyr_gill_lenc);
init_vector nsamp_gill_lenc(styr_gill_lenc,endyr_gill_lenc);
init_vector neff_gill_lenc(styr_gill_lenc,endyr_gill_lenc);
init_matrix obs_gill_lenc(styr_gill_lenc,endyr_gill_lenc,1,nlenbins);

//--><>--><>--><>-- Commercial Reduction fishery (also includes bait and recreational) --><>--><>--><>--><>--><>--><>--><>
// 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 styr_cR_agec;
init_int endyr_cR_agec;
init_int nyr_cR_agec;
!!cout << "number years agec" << nyr_cR_agec << endl;

init_ivector yrs_cR_agec(1,nyr_cR_agec);
init_vector nsamp_cR_agec(styr_cR_agec,endyr_cR_agec);
init_vector neff_cR_agec(styr_cR_agec,endyr_cR_agec);
init_matrix obs_cR_agec(styr_cR_agec,endyr_cR_agec,1,nages);

//##############################################################################
//##################Parameter values and initial guesses ##################################
//Initial guesses of estimated selectivity parameters
init_number set_selpar_L50_cR;
init_number set_selpar_slope_cR;
init_number set_selpar_L502_cR;
init_number set_selpar_slope2_cR;

init_number set_selpar_L50_gill;
init_number set_selpar_slope_gill;
init_number set_selpar_L502_gill;
init_number set_selpar_slope2_gill;

init_number set_sel_age0_gill; //input in logit space
init_number set_sel_age1_gill;
init_number set_sel_age2_gill;
init_number set_sel_age3_gill;
init_number set_sel_age4_gill;

init_number set_sel_age0_cR1; //input in logit space
init_number set_sel_age1_cR1;
init_number set_sel_age2_cR1;
init_number set_sel_age3_cR1;
init_number set_sel_age4_cR1;

init_number set_sel_age0_cR3; //input in logit space
init_number set_sel_age1_cR3;
init_number set_sel_age2_cR3;
init_number set_sel_age3_cR3;
init_number set_sel_age4_cR3;

init_number set_sel_age0_cR4; //input in logit space
init_number set_sel_age1_cR4;
init_number set_sel_age2_cR4;
init_number set_sel_age3_cR4;
init_number set_sel_age4_cR4;

//--weights for likelihood components--------------------------------------------------------------------------------
```

```
init_number set_w_L;
init_number set_w_ac;
init_number set_w_I_JAIs;       //JAI-seine
init_number set_w_I_JAIt;       //JAI-trawl
init_number set_w_I_gill;       //Adult index-gillnet
init_number set_w_gill_lenc;    //gillnet length comps
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_JAI_wgts;     //weight for penalty to keep JAI combination weights summing to 1.0

////--index catchability------------------------------------------------------------------------------------------------------------
init_number set_logq_JAIs;      //catchability coefficient (log) for seine JAI
init_number set_logq_JAIt;      //catchability coefficient (log) for trawl JAI
init_number set_logq_gill;      //catchability coefficient (log) for gillnet adult abundance

init_number set_JAI_exp;        //exponent for cpue index

//--JAI index combination weights------------------------------------------------------
init_number set_wgt_JAI1;
init_number set_wgt_JAI2;
init_number set_wgt_JAI3;
init_number set_wgt_JAI4;

//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_PN_var;     //assumed variance of RW q

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

//Tune Fapex (tuning removed in final year of optimization)
init_number set_Ftune;//not ok
init_int set_Ftune_yr;

//threshold sample sizes for length and age comps
init_number minSS_gill_lenc;
init_number minSS_cR_agec;

//switch to turn priors on off (-1 = off, 1 = on)
init_number switch_prior;

//ageing error matrix (columns are true ages, rows are ages as read for age comps)
init_matrix age_error(1,nages,1,nages);

//environmental factor (Mississippi River Flow)
init_vector env_fac(styr_cR_agec,endyr_cR_agec);

//switch to turn environmental factors  on/off in s-r function (1=on,2=off)
init_number switch_env_sr;
!!cout << switch_env_sr << endl;
//initial guess of s-r beta for environmental factors
init_number set_sr_beta_env;

//lengths at age and cv from reduction fishery to use for age-length conversions
init_vector set_length_age(1,nages);
init_vector set_len_cv(1,nages);
init_vector set_len_cv_se(1,nages);

//Von Bert parameters in TL mm
init_number set_Linf;
init_number set_K;
init_number set_t0;
init_number set_Linf_se;
init_number set_K_se;
init_number set_t0_se;


// #######Indexing integers for year(iyear), age(iage) ###############
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)
   {
     for(iyear=1; iyear<=1000; iyear++)
     {
       cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
```

```
       cout << "" <<endl;
     }
   }
   else
   {
    cout << "Data File read correctly" << endl;
   }
 END_CALCS

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

PARAMETER_SECTION
//////-----------------------------------------------------------------------------------------

  matrix wgt_fish_kg(styr,endyr,1,nages);
  matrix wgt_fish_mt(styr,endyr,1,nages);
  matrix wgt_spawn_kg(styr,endyr,1,nages);
  matrix wgt_spawn_mt(styr,endyr,1,nages);

  matrix wgt_cR_mt(styr,endyr,1,nages);      //wgt of cR landings in 1000 mt

  matrix lenprob(1,nages,1,nlenbins);           //distn of size at age (age-length key, 10 mm bins) in population
  matrix lenprob_plus(1,nages,1,nlenplus); //used to compute mass in last length bin (a plus group)
  matrix pred_gill_lenc(styr_gill_lenc,endyr_gill_lenc,1,nlenbins);

  init_bounded_vector len_cv(1,nages,0,0.8,-4);
  init_bounded_number Linf(150,1000,-2);  //Linf from VonB curve
  init_bounded_number K(0.1,0.8,-2);     //K from VonB curve
  init_bounded_number t0(-3,1,-2);       //t0 from VonB curve
  vector length_age(1,nages);           //vector of length at age

  vector nsamp_gill_lenc_allyr(styr_gill_lenc,endyr_gill_lenc);
  vector neff_gill_lenc_allyr(styr_gill_lenc,endyr_gill_lenc);
  vector neff_gill_lenc_allyr_out(styr_gill_lenc,endyr_gill_lenc);

  matrix pred_cR_agec(styr_cR_agec,endyr_cR_agec,1,nages);
  matrix ErrorFree_cR_agec(styr_cR_agec,endyr_cR_agec,1,nages); //age comps prior to applying ageing error matrix

  //nsamp_X_allyr vectors used only for R output of comps with nonconsecutive yrs, given sample size cutoffs
  vector nsamp_cR_agec_allyr(styr_cR_agec,endyr_cR_agec);

//effective sample size applied in multinomial distributions
  vector neff_cR_agec_allyr(styr_cR_agec,endyr_cR_agec);

//Computed effective sample size for output (not used in fitting)
  vector neff_cR_agec_allyr_out(styr_cR_agec,endyr_cR_agec);

//-----Population-----------------------------------------------------------------------------------
  matrix N(styr,endyr,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and cpue
  matrix N_spawn(styr,endyr,1,nages);        //Population numbers by year and age at peaking spawning: used for SSB
  matrix N_pred_agec(styr_cR_agec,endyr_cR_agec,1,nages);
  init_bounded_vector log_Nage_dev(2,nages,-5,5,1); //log deviations on initial abundance at age
  //vector log_Nage_dev(2,nages);
  vector log_Nage_dev_output(1,nages);              //used in output. equals zero for first age
  matrix B(styr,endyr,1,nages);           //Population biomass by year and age at start of yr
  vector totB(styr,endyr);                //Total biomass by year
  vector totN(styr,endyr);                //Total abundance by year
  vector SSB(styr,endyr);                   ///Total spawning biomass by year
  vector rec(styr,endyr);                 //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 maturity_m(1,nages);               //Proportion of male mature at age
  matrix reprod(styr,endyr,1,nages);
  vector wgted_reprod(1,nages);             //average reprod in last few years
//
////---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(1,20,1);      //log(virgin Recruitment)
  //number log_R0;
  number R0;                                //virgin recruitment
  init_bounded_number steep(0.21,0.99,3);   //steepness
  //number steep;  //uncomment to fix steepness, comment line directly above
  init_bounded_number rec_sigma(0.1,1.5,4);  //sd recruitment residuals
  number rec_sigma_sq;                       //square of rec_sigma
  number rec_logL_add;                       //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr,-15,15,2); //log recruitment deviations
  //vector log_rec_dev(styr_rec_dev,endyr);
  vector log_rec_dev_output(styr,endyr+1);  //used in output. equals zero except for yrs in log_rec_dev
  number var_rec_dev;                       //variance of log recruitment deviations
                                             //Estimate from yrs with unconstrainted S-R(XXXX-XXXX)
  number sigma_rec_dev;                     //sample SD of log residuals (may not equal rec_sigma
  init_bounded_number sr_beta_env(-10,20,-4);  //beta for environmental factor on stock-recruit function

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

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

//Commercial reduction-------------------------------------------------
  matrix sel_cR(styr,endyr,1,nages);
  init_bounded_number selpar_slope_cR1(0.5,10.0,-1); //period 1
  init_bounded_number selpar_L50_cR1(0.5,4.0,-1);
  init_bounded_number selpar_slope2_cR1(0.0,10.0,-2); //period 1
  init_bounded_number selpar_L502_cR1(0.0,6.0,-2);

  //init_bounded_number selpar_slope_cR2(0.5,10.0,-2); //period 2
  //init_bounded_number selpar_L50_cR2(0.5,4.0,-2);
```

```
  //init_bounded_number selpar_slope2_cR2(0.0,10.0,-3); //period 2
  //init_bounded_number selpar_L502_cR2(0.0,6.0,-3);
  //vector sel_cR2_vec(1,nages);

  init_bounded_number selpar_slope_cR3(0.5,10.0,-2); //period 3
  init_bounded_number selpar_L50_cR3(0.5,4.0,-2);
  init_bounded_number selpar_slope2_cR3(0.0,10.0,-3); //period 3
  init_bounded_number selpar_L502_cR3(0.0,6.0,-3);

  init_bounded_number selpar_slope_cR4(0.5,10.0,-3); //period 4
  init_bounded_number selpar_L50_cR4(0.5,4.0,-3);
  init_bounded_number selpar_slope2_cR4(0.0,10.0,-3); //period 4
  init_bounded_number selpar_L502_cR4(0.0,6.0,-3);

  init_bounded_vector sel_age0_cR1_logit(styr,endyr_period2,-15,15,-2);  //in logit space
  init_bounded_vector sel_age1_cR1_logit(styr,endyr_period2,-5,15,2);
  init_bounded_vector sel_age2_cR1_logit(styr,endyr_period2,-15,15,-2);
  init_bounded_vector sel_age3_cR1_logit(styr,endyr_period2,-5,15,2);
  init_bounded_vector sel_age4_cR1_logit(styr,endyr_period2,-5,15,2);
  vector sel_age_cR1_vec(1,nages);
  vector selpar_age0_cR1(styr,endyr_period2);
  vector selpar_age1_cR1(styr,endyr_period2);
  vector selpar_age2_cR1(styr,endyr_period2);
  vector selpar_age3_cR1(styr,endyr_period2);
  vector selpar_age4_cR1(styr,endyr_period2);

  init_bounded_vector sel_age0_cR3_logit(endyr_period2+1,endyr_period3,-15,15,-3);  //in logit space
  init_bounded_vector sel_age1_cR3_logit(endyr_period2+1,endyr_period3,-15,15,3);
  init_bounded_vector sel_age2_cR3_logit(endyr_period2+1,endyr_period3,-15,15,-3);
  init_bounded_vector sel_age3_cR3_logit(endyr_period2+1,endyr_period3,-15,15,-3);
  init_bounded_vector sel_age4_cR3_logit(endyr_period2+1,endyr_period3,-15,15,-3);
  vector sel_age_cR3_vec(1,nages);
  vector selpar_age0_cR3(endyr_period2+1,endyr_period3);
  vector selpar_age1_cR3(endyr_period2+1,endyr_period3);
  vector selpar_age2_cR3(endyr_period2+1,endyr_period3);
  vector selpar_age3_cR3(endyr_period2+1,endyr_period3);
  vector selpar_age4_cR3(endyr_period2+1,endyr_period3);

  init_bounded_vector sel_age0_cR4_logit(endyr_period3+1,endyr,-15,15,-3);  //in logit space
  init_bounded_vector sel_age1_cR4_logit(endyr_period3+1,endyr,-15,15,3);
  init_bounded_vector sel_age2_cR4_logit(endyr_period3+1,endyr,-15,15,-3);
  init_bounded_vector sel_age3_cR4_logit(endyr_period3+1,endyr,-15,15,-3);
  init_bounded_vector sel_age4_cR4_logit(endyr_period3+1,endyr,-15,15,-3);
  vector sel_age_cR4_vec(1,nages);
  vector selpar_age0_cR4(endyr_period3+1,endyr);
  vector selpar_age1_cR4(endyr_period3+1,endyr);
  vector selpar_age2_cR4(endyr_period3+1,endyr);
  vector selpar_age3_cR4(endyr_period3+1,endyr);
  vector selpar_age4_cR4(endyr_period3+1,endyr);

//Adult index from gillnet surveys-------------------------------------------
  matrix sel_gill(styr_gill_cpue,endyr_gill_cpue,1,nages);
  init_bounded_number selpar_slope_gill(0.5,20.0,-2); //period 1
  init_bounded_number selpar_L50_gill(0.0,4.0,-2);
  init_bounded_number selpar_slope2_gill(0.0,20.0,-3); //period 1
  init_bounded_number selpar_L502_gill(0.0,6.0,-3);

  init_bounded_number selpar_slope_gill2(0.5,10.0,-3); //period 2
  init_bounded_number selpar_L50_gill2(0.5,4.0,-3);
  init_bounded_number selpar_slope2_gill2(0.0,10.0,-4); //period 2
  init_bounded_number selpar_L502_gill2(0.0,6.0,-4);

  init_bounded_number sel_age0_gill_logit(-15,15,-3);  //in logit space
  init_bounded_number sel_age1_gill_logit(-15,15,3);
  init_bounded_number sel_age2_gill_logit(-15,15,-3);
  init_bounded_number sel_age3_gill_logit(-15,15,3);
  init_bounded_number sel_age4_gill_logit(-15,15,3);
  vector sel_age_gill_vec(1,nages);
  number selpar_age0_gill;
  number selpar_age1_gill;
  number selpar_age2_gill;
  number selpar_age3_gill;
  number selpar_age4_gill;

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

//-------CPUE Predictions--------------------------------
  vector obs_JAIs_cpue_final(styr_JAIs_cpue,endyr_JAIs_cpue);     //used to store cpue used in likelihood fit
  vector JAIs_cpue_cv_final(styr_JAIs_cpue,endyr_JAIs_cpue);
  vector pred_JAIs_cpue(styr_JAIs_cpue,endyr_JAIs_cpue);          //predicted JAI U for seine survey
  vector N_JAIs(styr_JAIs_cpue,endyr_JAIs_cpue);          //used to compute JAI index

  vector obs_JAIt_cpue_final(styr_JAIt_cpue,endyr_JAIt_cpue);     //used to store cpue used in likelihood fit
  vector JAIt_cpue_cv_final(styr_JAIt_cpue,endyr_JAIt_cpue);
  vector pred_JAIt_cpue(styr_JAIt_cpue,endyr_JAIt_cpue);          //predicted JAI U for trawl survey
  vector N_JAIt(styr_JAIt_cpue,endyr_JAIt_cpue);          //used to compute JAI index

  vector pred_gill_cpue(styr_gill_cpue,endyr_gill_cpue);          //predicted gillnet U
  matrix N_gill(styr_gill_lenc,endyr_gill_lenc,1,nages);          //used to compute gillnet index

//------Index exponent---------------------------------
  init_bounded_number JAI_exp(0.01,1.0,-3);

//------Index combination weights---------------------------------
  init_bounded_number wgt_JAI1(0.001,1.0,-3);
  init_bounded_number wgt_JAI2(0.001,1.0,-3);
  init_bounded_number wgt_JAI3(0.001,1.0,-3);
  init_bounded_number wgt_JAI4(0.001,1.0,-3);
  number JAI_wgt_sum_constraint;

////---Catchability (CPUE q's)-----------------------------------------------------------
  init_bounded_number log_q_JAIs(-20,10,1);    //seine
```

```
  init_bounded_number log_q_JAIt(-20,-5,-1);    //trawl
  init_bounded_number log_q_gill(-20,10,1);    //gillnet
  init_bounded_number q_rate(0.001,0.1,set_q_rate_phase);
  //number q_rate;
  //vector q_rate_fcn_PN(styr_PN_cpue,endyr_PN_cpue);  //increase due to technology creep (saturates in 2003)

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

  //init_bounded_vector q_RW_log_dev_gill(styr_gill_cpue,endyr_gill_cpue-1,-3.0,3.0,set_q_RW_phase);
  //vector q_gill(styr_gill_cpue,endyr_gill_cpue);

//---Landings in numbers (total or 1000 fish) and in wgt (klb)--------------------------------------------------
  matrix L_cR_num(styr,endyr,1,nages);             //landings (numbers) at age
  matrix L_cR_mt(styr,endyr,1,nages);             //landings (1000 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 1000 mt summed over ages
  matrix L_cR_num_agec(styr_cR_agec,endyr_cR_agec,1,nages);

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

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

  number R_med;                                //median recruitment for chosen benchmark years
  vector R_temp(styr_bench,endyr_bench);
  vector R_sort(styr_bench,endyr_bench);
  number SPR_med;                              //median SSB/R (R = SSB year+1) for chosen SSB years
  number SPR_75th;
  vector SPR_temp(styr_bench,endyr_bench);
  vector SPR_sort(styr_bench,endyr_bench);
  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;

////---MSY calcs----------------------------------------------------------------------------
  number F_cR_prop;       //proportion of F_sum attributable to reduction, last X=selpar_n_yrs_wgted yrs, used for avg body weights
  number F_temp_sum;      //sum of geom mean Fsum's in last X yrs, used to compute F_fishery_prop

  vector F_end(1,nages);
  vector F_end_L(1,nages);
  number F_end_apex;

  number SSB_msy_out;           //SSB (total mature biomass) at msy
  number F_msy_out;             //F at msy
  number msy_mt_out;           //max sustainable yield (1000 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(1000 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 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);
  number SdSSB_msy_end;
  number FdF_msy_end;
  number FdF_msy_end_mean;         //geometric mean of last 3 years

  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)

////--------Mortality------------------------------------------------------------------
  vector M(1,nages);                         //age-dependent natural mortality
  number M_constant;                          //age-indpendent: used only for MSST
  matrix M_mat(styr,endyr,1,nages);
  vector wgted_M(1,nages);                   //weighted M vector for last few years

  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
  matrix Z(styr,endyr,1,nages);

  vector E(styr,endyr);                     //Exploitation rate
  vector F_age2plus(styr,endyr);            //population weighted age 2+ F
  vector F_cR_age2plus(styr,endyr);         //population weighted age 2+ F
```

```
  init_bounded_number log_avg_F_cR(-10,5.0,1);
  init_bounded_dev_vector log_F_dev_cR(styr_cR_L,endyr_cR_L,-10.0,10.0,2);
  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;

  init_bounded_number F_init_ratio(0.05,1.5,-3);

//---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 F_spr_age2plus(1,n_iter_spr);  //values of F age2+ 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
  vector spr_F0(styr,endyr);         //Spawning biomass per recruit at F=0
  vector bpr_F0(styr,endyr);         //Biomass per recruit at F=0
  number wgted_spr_F0;

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


////-------Objective function components-----------------------------------------------------------------------------
  number w_L;
  number w_ac;
  number w_I_JAIs;
  number w_I_JAIt;
  number w_I_gill;
  number w_I_gill_lc;
  number w_rec;
  number w_rec_early;
  number w_rec_end;
  number w_fullF;
  number w_Ftune;
  number w_JAI_wgts;

  number f_JAIs_cpue;
  number f_JAIt_cpue;
  number f_gill_cpue;

  number f_cR_L;

  number f_cR_agec;

  number f_gill_lenc;

  //number f_PN_RW_cpue; //random walk component of indices

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

  number f_priors;                  //prior information on parameters

  objective_function_value fval;
  number fval_unwgt;

//--Dummy variables ----
  number denom;                   //denominator used in some calculations
  number numer;                   //numerator used in some calculations
  number numer1;
  number denom1;
  vector temp_agevec(1,nages);
  number dum1;

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


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

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
RUNTIME_SECTION
 maximum_function_evaluations 1000, 4000,8000, 10000;
 convergence_criteria 1e-2, 1e-5,1e-6, 1e-7;

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

// Set values of fixed parameters or set initial guess of estimated parameters
  M=set_M;
```

```
M_constant=set_M_constant;
M_mat=set_M_mat;

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

log_q_JAIs=set_logq_JAIs;
log_q_JAIt=set_logq_JAIt;
log_q_gill=set_logq_gill;

JAI_exp=set_JAI_exp;

wgt_JAI1=set_wgt_JAI1;
wgt_JAI2=set_wgt_JAI2;
wgt_JAI3=set_wgt_JAI3;
wgt_JAI4=set_wgt_JAI4;

q_rate=set_q_rate;
//q_rate_fcn_PN=1.0;
q_DD_beta=set_q_DD_beta;
q_DD_fcn=1.0;
//q_RW_log_dev_gill.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_cL(iyear)=(1.0+q_rate)*q_rate_fcn_cL(iyear-1); //compound
//           q_rate_fcn_PN(iyear)=(1.0+(iyear-styr_PN_cpue)*q_rate)*q_rate_fcn_PN(styr_PN_cpue);  //linear
//        }
//        if (iyear>2003) {q_rate_fcn_PN(iyear)=q_rate_fcn_PN(iyear-1);}
//    }
//} //end q_rate conditional

w_L=set_w_L;
w_ac=set_w_ac;
w_I_JAIs=set_w_I_JAIs;
w_I_JAIt=set_w_I_JAIt;
w_I_gill=set_w_I_gill;
w_I_gill_lc=set_w_gill_lenc;
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_JAI_wgts=set_w_JAI_wgts;

log_avg_F_cR=set_log_avg_F_cR;
F_init_ratio=set_F_init_ratio;

log_R0=set_log_R0;

length_age=set_length_age;
len_cv=set_len_cv;
Linf=set_Linf;
K=set_K;
t0=set_t0;

selpar_L50_cR1=set_selpar_L50_cR;
selpar_slope_cR1=set_selpar_slope_cR;
selpar_L502_cR1=set_selpar_L502_cR;
selpar_slope2_cR1=set_selpar_slope2_cR;

//selpar_L50_cR2=set_selpar_L50_cR;
//selpar_slope_cR2=set_selpar_slope_cR;
//selpar_L502_cR2=set_selpar_L502_cR;
//selpar_slope2_cR2=set_selpar_slope2_cR;

selpar_L50_cR3=set_selpar_L50_cR;
selpar_slope_cR3=set_selpar_slope_cR;
selpar_L502_cR3=set_selpar_L502_cR;
selpar_slope2_cR3=set_selpar_slope2_cR;

selpar_L50_cR4=set_selpar_L50_cR;
selpar_slope_cR4=set_selpar_slope_cR;
selpar_L502_cR4=set_selpar_L502_cR;
selpar_slope2_cR4=set_selpar_slope2_cR;

selpar_L50_gill=set_selpar_L50_gill;
selpar_slope_gill=set_selpar_slope_gill;
selpar_L502_gill=set_selpar_L502_gill;
selpar_slope2_gill=set_selpar_slope2_gill;

selpar_L50_gill2=set_selpar_L50_gill;
selpar_slope_gill2=set_selpar_slope_gill;
selpar_L502_gill2=set_selpar_L502_gill;
selpar_slope2_gill2=set_selpar_slope2_gill;

sel_age0_gill_logit=set_sel_age0_gill;
sel_age1_gill_logit=set_sel_age1_gill;
sel_age2_gill_logit=set_sel_age2_gill;
sel_age3_gill_logit=set_sel_age3_gill;
sel_age4_gill_logit=set_sel_age4_gill;

for (iyear=styr; iyear<=endyr_period2; iyear++)
  {
   sel_age0_cR1_logit(iyear)=set_sel_age0_cR1;
   sel_age1_cR1_logit(iyear)=set_sel_age1_cR1;
   sel_age2_cR1_logit(iyear)=set_sel_age2_cR1;
   sel_age3_cR1_logit(iyear)=set_sel_age3_cR1;
   sel_age4_cR1_logit(iyear)=set_sel_age4_cR1;
  }
```

```
  for (iyear=endyr_period2+1; iyear<=endyr_period3; iyear++)
    {
     sel_age0_cR3_logit(iyear)=set_sel_age0_cR3;
     sel_age1_cR3_logit(iyear)=set_sel_age1_cR3;
     sel_age2_cR3_logit(iyear)=set_sel_age2_cR3;
     sel_age3_cR3_logit(iyear)=set_sel_age3_cR3;
     sel_age4_cR3_logit(iyear)=set_sel_age4_cR3;
    }
  for (iyear=endyr_period3+1; iyear<=endyr; iyear++)
    {
     sel_age0_cR4_logit(iyear)=set_sel_age0_cR4;
     sel_age1_cR4_logit(iyear)=set_sel_age1_cR4;
     sel_age2_cR4_logit(iyear)=set_sel_age2_cR4;
     sel_age3_cR4_logit(iyear)=set_sel_age3_cR4;
     sel_age4_cR4_logit(iyear)=set_sel_age4_cR4;
    }

 sqrt2pi=sqrt(2.*3.14159265);
 //g2mt=0.000001;         //conversion of grams to metric tons
 g2mt=1.0;
 g2kg=0.001;            //conversion of grams to kg
 mt2klb=2.20462;        //conversion of metric tons to 1000 lb
 mt2lb=mt2klb*1000.0;   //conversion of metric tons to lb
 g2klb=g2mt*mt2klb;     //conversion of grams to 1000 lb
 dzero=0.00001;         //additive constant to prevent division by zero
 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;
 maturity_m=maturity_m_obs;
 prop_f=prop_f_obs;

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

      nsamp_cR_agec_allyr=missing;

      neff_cR_agec_allyr=missing;

      for (iyear=styr_cR_agec; iyear<=endyr_cR_agec; iyear++)
         {
           if (nsamp_cR_agec(iyear)>=minSS_cR_agec)
           {
             nsamp_cR_agec_allyr(iyear)=nsamp_cR_agec(iyear);
             neff_cR_agec_allyr(iyear)=neff_cR_agec(iyear);
           }
         }

      //cout << "nsamp_cR_agec" << nsamp_cR_agec_allyr << endl;

      nsamp_gill_lenc_allyr=missing;

      neff_gill_lenc_allyr=missing;

      for (iyear=styr_gill_lenc; iyear<=endyr_gill_lenc; iyear++)
         {
           if (nsamp_gill_lenc(iyear)>=minSS_gill_lenc)
           {
             nsamp_gill_lenc_allyr(iyear)=nsamp_gill_lenc(iyear);
             neff_gill_lenc_allyr(iyear)=neff_gill_lenc(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();

  L_total_knum_yr.initialize();
  L_total_mt_yr.initialize();

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


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

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

 R0=mfexp(log_R0);

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

FUNCTION get_weight_at_age
  //compute mean length (mm) and weight (whole) at age
    length_age=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));
    wgt_fish_kg=g2kg*wgt_fish_g;     //wgt in kilograms
    wgt_fish_mt=g2mt*wgt_fish_g;     //mt of whole wgt: g2mt converts g to mt
    wgt_spawn_kg=g2kg*wgt_spawn_g;   //wgt in kilograms
    wgt_spawn_mt=g2mt*wgt_spawn_g;   //mt of whole wgt: g2mt converts g to mt

FUNCTION get_reprod

  //product of stuff going into reproductive capacity calcs
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    //reprod(iyear)=elem_prod((elem_prod(prop_f,maturity_f)+elem_prod((1.0-prop_f),maturity_m)),wgt_spawn_mt(iyear));
    //reprod(iyear)=elem_prod((elem_prod(prop_f,maturity_f)+elem_prod((1.0-prop_f),maturity_m)),fec_eggs(iyear));
    reprod(iyear)=elem_prod(elem_prod(prop_f,maturity_f),fec_eggs(iyear));
  }

  //compute average natural mortality
  wgted_M=M_mat(endyr)*0.0;
  for(iyear=(endyr-selpar_n_yrs_wgted+1); iyear<=endyr; iyear++)
  {
    wgted_M+=M_mat(iyear);
  }
  wgted_M=wgted_M/selpar_n_yrs_wgted;

  //average reprod for last few years for eq calculations
  wgted_reprod=reprod(endyr)*0.0;
  for(iyear=(endyr-selpar_n_yrs_wgted+1); iyear<=endyr; iyear++)
  {
    wgted_reprod+=reprod(iyear);
  }
  wgted_reprod=wgted_reprod/selpar_n_yrs_wgted;


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

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

 //cout << "lenprob" << lenprob << endl;


FUNCTION get_weight_at_age_landings

  wgt_cR_mt=wgt_fish_mt;

FUNCTION get_spr_F0

  for (iyear=styr; iyear<=endyr; iyear++)
```

```
  {
    //at mdyr, apply half this yr’s mortality, half next yr’s
    N_spr_F0(1)=1.0*mfexp(-1.0*M_mat(iyear,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)));
      dum1=M_mat(iyear,iage-1)*(1.0-spawn_time_frac) + M_mat(iyear,iage)*spawn_time_frac;
      N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(dum1));
      N_bpr_F0(iage)=N_bpr_F0(iage-1)*mfexp(-1.0*(M_mat(iyear,iage-1)));
    }
    N_spr_F0(nages)=N_spr_F0(nages)/(1.0-mfexp(-1.0*M_mat(iyear,nages))); //plus group (sum of geometric series)
    N_bpr_F0(nages)=N_bpr_F0(nages)/(1.0-mfexp(-1.0*M_mat(iyear,nages)));

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

  N_spr_F0(1)=1.0*mfexp(-1.0*wgted_M(1)*spawn_time_frac); //at peak spawning time
  for (iage=2; iage<=nages; iage++)
  {
    dum1=wgted_M(iage-1)*(1.0-spawn_time_frac) + wgted_M(iage)*spawn_time_frac;
    N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(dum1));
  }
  N_spr_F0(nages)=N_spr_F0(nages)/(1.0-mfexp(-1.0*wgted_M(nages))); //plus group (sum of geometric series
  wgted_spr_F0=sum(elem_prod(N_spr_F0,wgted_reprod));

FUNCTION get_selectivity

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

  //gillnet survey selectivity
  selpar_age0_gill=1.0/(1.0+mfexp(-sel_age0_gill_logit));
  selpar_age1_gill=1.0/(1.0+mfexp(-sel_age1_gill_logit));
  selpar_age2_gill=1.0/(1.0+mfexp(-sel_age2_gill_logit));
  selpar_age3_gill=1.0/(1.0+mfexp(-sel_age3_gill_logit));
  selpar_age4_gill=1.0/(1.0+mfexp(-sel_age4_gill_logit));
  sel_age_gill_vec(1)=selpar_age0_gill;
  sel_age_gill_vec(2)=selpar_age1_gill;
  sel_age_gill_vec(3)=selpar_age2_gill;
  sel_age_gill_vec(4)=selpar_age3_gill;
  sel_age_gill_vec(5)=selpar_age4_gill;
  //sel_age_gill_vec=sel_age_gill_vec/max(sel_age_gill_vec); //to scale to one
  for (iyear=styr_gill_cpue; iyear<=endyr_period1_gill; iyear++)
  //for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
  {  //time-invariant selectivities
     //sel_gill(iyear)=logistic(agebins, selpar_L50_gill, selpar_slope_gill);
     //sel_gill(iyear)=logistic_double(agebins,selpar_L50_gill,selpar_slope_gill,selpar_L502_gill,selpar_slope2_gill);
     sel_gill(iyear)=sel_age_gill_vec;
  }

  //cout << "end_yrp1" << endyr_period1 << endl;

  for (iyear=endyr_period1_gill+1; iyear<=endyr_gill_cpue; iyear++)
  {  //time-invariant selectivities
     sel_gill(iyear)=sel_gill(styr_gill_cpue);
     //sel_gill(iyear)=logistic(agebins, selpar_L50_gill2, selpar_slope_gill2);
     //sel_gill(iyear)=logistic_double(agebins,selpar_L50_gill2,selpar_slope_gill2,selpar_L502_gill2,selpar_slope2_gill2);
  }


  //commercial reduction selectivity
  //Period 1:
  for (iyear=styr; iyear<=endyr_period2; iyear++)
   {
    if (iyear>endyr_period1) {selpar_age0_cR1(iyear)=1.0/(1.0+mfexp(-sel_age0_cR1_logit(iyear)));
      selpar_age1_cR1(iyear)=1.0/(1.0+mfexp(-sel_age1_cR1_logit(iyear)));
      selpar_age2_cR1(iyear)=1.0/(1.0+mfexp(-sel_age2_cR1_logit(iyear)));
      selpar_age3_cR1(iyear)=1.0/(1.0+mfexp(-sel_age3_cR1_logit(iyear)));
      selpar_age4_cR1(iyear)=1.0/(1.0+mfexp(-sel_age4_cR1_logit(iyear)));
      sel_age_cR1_vec(1)=selpar_age0_cR1(iyear);
      sel_age_cR1_vec(2)=selpar_age1_cR1(iyear);
      sel_age_cR1_vec(3)=selpar_age2_cR1(iyear);
      sel_age_cR1_vec(4)=selpar_age3_cR1(iyear);
      sel_age_cR1_vec(5)=selpar_age4_cR1(iyear);}
     else {selpar_age0_cR1(iyear)=1.0/(1.0+
          (mfexp(-sel_age0_cR1_logit(endyr_period1+1))+
           mfexp(-sel_age0_cR1_logit(endyr_period1+2))+
           mfexp(-sel_age0_cR1_logit(endyr_period1+3)))/3);
      selpar_age1_cR1(iyear)=1.0/(1.0+
          (mfexp(-sel_age1_cR1_logit(endyr_period1+1))+
           mfexp(-sel_age1_cR1_logit(endyr_period1+2))+
           mfexp(-sel_age1_cR1_logit(endyr_period1+3)))/3);
      selpar_age2_cR1(iyear)=1.0/(1.0+
          (mfexp(-sel_age2_cR1_logit(endyr_period1+1))+
           mfexp(-sel_age2_cR1_logit(endyr_period1+2))+
           mfexp(-sel_age2_cR1_logit(endyr_period1+3)))/3);
      selpar_age3_cR1(iyear)=1.0/(1.0+
          (mfexp(-sel_age3_cR1_logit(endyr_period1+1))+
           mfexp(-sel_age3_cR1_logit(endyr_period1+2))+
           mfexp(-sel_age3_cR1_logit(endyr_period1+3)))/3);
      selpar_age4_cR1(iyear)=1.0/(1.0+
          (mfexp(-sel_age4_cR1_logit(endyr_period1+1))+
           mfexp(-sel_age4_cR1_logit(endyr_period1+2))+
           mfexp(-sel_age4_cR1_logit(endyr_period1+3)))/3);
      sel_age_cR1_vec(1)=selpar_age0_cR1(iyear);
      sel_age_cR1_vec(2)=selpar_age1_cR1(iyear);
      sel_age_cR1_vec(3)=selpar_age2_cR1(iyear);
      sel_age_cR1_vec(4)=selpar_age3_cR1(iyear);
      sel_age_cR1_vec(5)=selpar_age4_cR1(iyear);}
     //sel_age_cR1_vec=sel_age_cR1_vec/max(sel_age_cR1_vec);
     //sel_age_cR1_vec=sel_age_cR1_vec/max(sel_age_cR1_vec); //to scale to one
     //sel_cR(iyear)=logistic(agebins,selpar_L50_cR1,selpar_slope_cR1);
     //sel_cR(iyear)=logistic_double(agebins,selpar_L50_cR1,selpar_slope_cR1,selpar_L502_cR1,selpar_slope2_cR1);
```

```
    sel_cR(iyear)=sel_age_cR1_vec;
  }

  //Period 2:
  //for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  //{
  //   sel_cR(iyear)=sel_cR(styr);
  //}

  //Period 3
  for (iyear=endyr_period2+1; iyear<=endyr_period3; iyear++)
   {
    selpar_age0_cR3(iyear)=1.0/(1.0+mfexp(-sel_age0_cR3_logit(iyear)));
    selpar_age1_cR3(iyear)=1.0/(1.0+mfexp(-sel_age1_cR3_logit(iyear)));
    selpar_age2_cR3(iyear)=1.0/(1.0+mfexp(-sel_age2_cR3_logit(iyear)));
    selpar_age3_cR3(iyear)=1.0/(1.0+mfexp(-sel_age3_cR3_logit(iyear)));
    selpar_age4_cR3(iyear)=1.0/(1.0+mfexp(-sel_age4_cR3_logit(iyear)));
    sel_age_cR3_vec(1)=selpar_age0_cR3(iyear);
    sel_age_cR3_vec(2)=selpar_age1_cR3(iyear);
    sel_age_cR3_vec(3)=selpar_age2_cR3(iyear);
    sel_age_cR3_vec(4)=selpar_age3_cR3(iyear);
    sel_age_cR3_vec(5)=selpar_age4_cR3(iyear);
    //sel_age_cR3_vec=sel_age_cR3_vec/max(sel_age_cR3_vec); //to scale to one
     //sel_cR(iyear)=sel_cR(styr);
     //sel_cR(iyear)=logistic(agebins,selpar_L50_cR3,selpar_slope_cR3);
     //sel_cR(iyear)=logistic_double(agebins,selpar_L50_cR3,selpar_slope_cR3,selpar_L502_cR3,selpar_slope2_cR3);
     sel_cR(iyear)=sel_age_cR3_vec;
  }

  //Period 4
  for (iyear=endyr_period3+1; iyear<=endyr; iyear++)
   {
    selpar_age0_cR4(iyear)=1.0/(1.0+mfexp(-sel_age0_cR4_logit(iyear)));
    selpar_age1_cR4(iyear)=1.0/(1.0+mfexp(-sel_age1_cR4_logit(iyear)));
    selpar_age2_cR4(iyear)=1.0/(1.0+mfexp(-sel_age2_cR4_logit(iyear)));
    selpar_age3_cR4(iyear)=1.0/(1.0+mfexp(-sel_age3_cR4_logit(iyear)));
    selpar_age4_cR4(iyear)=1.0/(1.0+mfexp(-sel_age4_cR4_logit(iyear)));
    sel_age_cR4_vec(1)=selpar_age0_cR4(iyear);
    sel_age_cR4_vec(2)=selpar_age1_cR4(iyear);
    sel_age_cR4_vec(3)=selpar_age2_cR4(iyear);
    sel_age_cR4_vec(4)=selpar_age3_cR4(iyear);
    sel_age_cR4_vec(5)=selpar_age4_cR4(iyear);
    //sel_age_cR4_vec=sel_age_cR4_vec/max(sel_age_cR4_vec); //to scale to one
     //sel_cR(iyear)=logistic(agebins, selpar_L50_cR4,selpar_slope_cR4);
     //sel_cR(iyear)=logistic_double(agebins,selpar_L50_cR4,selpar_slope_cR4,selpar_L502_cR4,selpar_slope2_cR4);
     //sel_cR(iyear)=sel_cR(styr);
     sel_cR(iyear)=sel_age_cR4_vec;
  }

FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  ////initialization F is avg of 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));}
    if (iyear<styr_cR_L)
      {F_cR_out(iyear)=mfexp(log_avg_F_cR+log_F_dev_init_cR);}
    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_mat(iyear)+F(iyear);
  }  //end iyear

FUNCTION get_bias_corr
  //may exclude last BiasCor_exclude_yrs yrs bc constrained or lack info to estimate
  //var_rec_dev=norm2(log_rec_dev(styr_rec_dev,(endyr-BiasCor_exclude_yrs))-
  //            sum(log_rec_dev(styr_rec_dev,(endyr-BiasCor_exclude_yrs)))
  //            /(nyrs_rec-BiasCor_exclude_yrs))/(nyrs_rec-BiasCor_exclude_yrs-1.0);
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_phase2)-
              sum(log_rec_dev(styr_rec_dev,endyr_rec_phase2))
              /(nyrs_rec-(endyr_rec_phase2-styr_rec_dev)))/(nyrs_rec-(endyr_rec_phase2-styr_rec_dev)-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(styr)*R0;

  if(set_SR_switch>1) //Beverton-Holt
  {
    R_virgin=(R0/((5.0*steep-1.0)*spr_F0(styr)))*
                 (BiasCor*4.0*steep*spr_F0(styr)-spr_F0(styr)*(1.0-steep));
  }
  if(set_SR_switch<2) //Ricker
  {
    R_virgin=R0/spr_F0(styr)*(1+log(BiasCor*spr_F0(styr))/steep);
  }

  B0=bpr_F0(styr)*R_virgin;
  //temp_agevec=wgt_fish_mt(styr);
  //B0_q_DD=R_virgin*sum(elem_prod(N_bpr_F0(set_q_DD_stage,nages),temp_agevec(set_q_DD_stage,nages)));
```

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


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

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

//with environmental factor
  if(switch_env_sr=1)
  {
  if(set_SR_switch>1) //Beverton-Holt
  {
    if (styr=styr_rec_dev) {R1=((R0/((5.0*steep-1.0)*spr_initial))*
                 (4.0*steep*spr_initial-spr_F0(styr)*(1.0-steep)))
                 *mfexp(sr_beta_env*env_fac(styr));} //without bias correction (deviation added later)
    else {R1=((R0/((5.0*steep-1.0)*spr_initial))*
                 (BiasCor*4.0*steep*spr_initial-spr_F0(styr)*(1.0-steep)))
                 *mfexp(sr_beta_env*env_fac(styr));} //with bias correction
  }
  if(set_SR_switch<2) //Ricker
  {
    if (styr=styr_rec_dev) {R1=(R0/spr_initial*(1+log(spr_initial/steep)))
                            *mfexp(sr_beta_env*env_fac(styr));} //without bias correction (deviation added later)
    else {R1=(R0/spr_initial*(1+log(BiasCor*spr_initial)/steep))
                *mfexp(sr_beta_env*env_fac(styr));} //with bias correction
  }
  }

//without environmental factor
 if(switch_env_sr=2)
 {
 if(set_SR_switch>1) //Beverton-Holt
  {
    if (styr=styr_rec_dev) {R1=(R0/((5.0*steep-1.0)*spr_initial))*
                 (4.0*steep*spr_initial-spr_F0(styr)*(1.0-steep));} //without bias correction (deviation added later)
    else {R1=(R0/((5.0*steep-1.0)*spr_initial))*
                 (BiasCor*4.0*steep*spr_initial-spr_F0(styr)*(1.0-steep));} //with bias correction
  }
  if(set_SR_switch<2) //Ricker
  {
    if (styr=styr_rec_dev) {R1=R0/spr_initial*(1+log(spr_initial/steep));} //without bias correction (deviation added later)
    else {R1=R0/spr_initial*(1+log(BiasCor*spr_initial)/steep);} //with bias correction
  }
 }

  if(R1<0.0) {R1=10.0;} //Avoid negative popn sizes during search algorithm


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

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

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

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

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

//Rest of years
  for (iyear=styr; iyear<endyr; iyear++)
  {
    if(iyear<(styr_rec_dev-1)) //recruitment follows S-R curve exactly
    {
        N(iyear+1,1)=0.0;
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod(iyear+1)));
        temp_agevec=wgt_fish_mt(iyear+1);
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),temp_agevec(set_q_DD_stage,nages)));
        //add dzero to avoid log(zero)
        //with environmental factor
        if(switch_env_sr=1)
        {
        if(set_SR_switch>1) //Beverton-Holt
        {
          N(iyear+1,1)=(BiasCor*mfexp(log(((0.8*R0*steep*SSB(iyear+1))/(0.2*R0*spr_F0(iyear+1)*
            (1.0-steep)+(steep-0.2)*SSB(iyear+1)))+dzero)))
            *mfexp(sr_beta_env*env_fac(iyear+1)); //Vaughan et al 2011
        }
        if(set_SR_switch<2) //Ricker
```

```
        {
          N(iyear+1,1)=(mfexp(log(BiasCor*SSB(iyear+1)/spr_F0(iyear)*mfexp(steep*(1-SSB(iyear+1)/(R0*spr_F0(iyear+1))))+dzero)))
                        *mfexp(sr_beta_env*env_fac(iyear+1));
        }
        }
        //without environmenal factor
        if(switch_env_sr=2)
        {
        if(set_SR_switch>1) //Beverton-Holt
        {
          N(iyear+1,1)=BiasCor*mfexp(log(((0.8*R0*steep*SSB(iyear+1))/(0.2*R0*spr_F0(iyear+1)*
            (1.0-steep)+(steep-0.2)*SSB(iyear+1)))+dzero));
        }
        if(set_SR_switch<2) //Ricker
        {
          N(iyear+1,1)=mfexp(log(BiasCor*SSB(iyear+1)/spr_F0(iyear+1)*mfexp(steep*(1-SSB(iyear+1)/(R0*spr_F0(iyear+1))))+dzero));
        }
        }
    }
    else   //recruitment follows S-R curve with lognormal deviation
    {
        N(iyear+1,1)=0.0;
        N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
        N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
        N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
        N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
        SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod(iyear+1)));
        temp_agevec=wgt_fish_mt(iyear+1);
        B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),temp_agevec(set_q_DD_stage,nages)));
        //add dzero to avoid log(zero)
        //with environmental factor
        if(switch_env_sr=1)
        {
        if(set_SR_switch>1) //Beverton-Holt
        {
          N(iyear+1,1)=(mfexp(log(((0.8*R0*steep*SSB(iyear+1))/(0.2*R0*spr_F0(iyear+1)*
            (1.0-steep)+(steep-0.2)*SSB(iyear+1)))+dzero)+log_rec_dev(iyear+1)))*mfexp(sr_beta_env*env_fac(iyear+1));
        }
        if(set_SR_switch<2) //Ricker
        {
          N(iyear+1,1)=(mfexp(log(SSB(iyear+1)/spr_F0(iyear+1)*mfexp(steep*(1-SSB(iyear+1)/(R0*spr_F0(iyear+1))))+dzero)+log_rec_dev(iyear+1)))
                         *mfexp(sr_beta_env*env_fac(iyear+1));
        }
        }
        //without environmental factor
        if(switch_env_sr=2)
        {
        if(set_SR_switch>1) //Beverton-Holt
        {
          N(iyear+1,1)=mfexp(log(((0.8*R0*steep*SSB(iyear+1))/(0.2*R0*spr_F0(iyear+1)*
            (1.0-steep)+(steep-0.2)*SSB(iyear+1)))+dzero)+log_rec_dev(iyear+1));
        }
        if(set_SR_switch<2) //Ricker
        {
          N(iyear+1,1)=mfexp(log(SSB(iyear+1)/spr_F0(iyear+1)*mfexp(steep*(1-SSB(iyear+1)/(R0*spr_F0(iyear+1))))+dzero)+log_rec_dev(iyear+1));
        }
        }
        }
  }
  //cout << "N" << N << endl;
  //cout << "R0" << R0 << endl;

    //last year (projection) has no recruitment variability
    //N(endyr+1,1)=0.0;
    //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
    //if(set_SR_switch>1) //Beverton-Holt
   // {
    //  N(endyr+1,1)=mfexp(log(((0.8*R0*steep*SSB(endyr))/(0.2*R0*spr_F0(endyr)*
    //             (1.0-steep)+(steep-0.2)*SSB(endyr)))+dzero));
    //}
    ///if(set_SR_switch<2) //Ricker
    //{
    //  N(endyr+1,1)=mfexp(log(SSB(endyr+1)/spr_F0(endyr)*mfexp(steep*(1-SSB(endyr+1)/(R0*spr_F0(endyr))))+dzero));
    //}

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

  SdS0=SSB/S0; //trillions of eggs/eggs
   //cout << "SDS0" << SdS0 << endl;
  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));
  }


FUNCTION get_landings_wgt

////---Predicted landings------------------------
  for (iyear=styr; iyear<=endyr; iyear++)
```

```
  {
    L_cR_mt(iyear)=elem_prod(L_cR_num(iyear),wgt_cR_mt(iyear)); //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_PN_cpue; iyear<=endyr_PN_cpue; iyear++)
 //     {   if (iyear>styr_PN_cpue & iyear <=2003)
 //         {//q_rate_fcn_cL(iyear)=(1.0+q_rate)*q_rate_fcn_cL(iyear-1); //compound
 //            q_rate_fcn_PN(iyear)=(1.0+(iyear-styr_PN_cpue)*q_rate)*q_rate_fcn_PN(styr_PN_cpue);  //linear
 //         }
 //         if (iyear>2003) {q_rate_fcn_PN(iyear)=q_rate_fcn_PN(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------------------------
 //combined JAI index
  if(JAI_cpue_switch==1)
  {
    obs_JAIs_cpue_final=pow(obs_JAIs_cpue,JAI_exp);
    JAIs_cpue_cv_final=JAIs_cpue_cv;
    obs_JAIt_cpue_final=pow(obs_JAIt_cpue,JAI_exp);
    JAIt_cpue_cv_final=JAIt_cpue_cv;
  }
  else
  {
    obs_JAIs_cpue_final=(obs_JAI1_cpue*wgt_JAI1+obs_JAI2_cpue*wgt_JAI2+obs_JAI3_cpue*wgt_JAI3+obs_JAI4_cpue*wgt_JAI4)
                       /(wgt_JAI1+wgt_JAI2+wgt_JAI3+wgt_JAI4);
    obs_JAIs_cpue_final=pow(obs_JAIs_cpue_final,JAI_exp);
    JAIs_cpue_cv_final=(JAI1_cpue_cv*wgt_JAI1+JAI2_cpue_cv*wgt_JAI2+JAI3_cpue_cv*wgt_JAI3+JAI4_cpue_cv*wgt_JAI4)
                       /(wgt_JAI1+wgt_JAI2+wgt_JAI3+wgt_JAI4);
  }

 //JAI seine  survey
  for (iyear=styr_JAIs_cpue; iyear<=endyr_JAIs_cpue; iyear++)
  {   //index in number units
      N_JAIs(iyear)=N(iyear,1);
      pred_JAIs_cpue(iyear)=mfexp(log_q_JAIs)*N_JAIs(iyear);
  }

 //JAI trawl  survey
  for (iyear=styr_JAIt_cpue; iyear<=endyr_JAIt_cpue; iyear++)
  {   //index in number units
      N_JAIt(iyear)=N(iyear,1);
      pred_JAIt_cpue(iyear)=mfexp(log_q_JAIt)*N_JAIt(iyear);
  }

 //Gillnet adult index
  for (iyear=styr_gill_cpue; iyear<=endyr_gill_cpue; iyear++)
  {   //index in number units
      N_gill(iyear)=elem_prod(N_mdyr(iyear),sel_gill(iyear));
      pred_gill_cpue(iyear)=mfexp(log_q_gill)*sum(N_gill(iyear));
  }

FUNCTION get_length_comps
 //Fishery independent

  //cout << "N_gill" << N_gill << endl;
  //cout << "lenprob" << lenprob << endl;
  //cout << "pred_gill_lenc" << pred_gill_lenc << endl;

  for (iyear=styr_gill_lenc;iyear<=endyr_gill_lenc;iyear++)
  {
    pred_gill_lenc(iyear)=(N_gill(iyear)*lenprob)/sum(N_gill(iyear));
  }
 // cout << "pred_gill_lenc" << pred_gill_lenc << endl;

FUNCTION get_age_comps

  //cout << "L_cR_num" << L_cR_num << endl;
  //cout << "yrs" << yrs_cR_agec << endl;

  for (iyear=styr_cR_agec;iyear<=endyr_cR_agec;iyear++)
  {
    L_cR_num_agec(iyear)=L_cR_num(iyear);
  }
  //cout << "L_cR_AGEC" << L_cR_num_agec << endl;

  //Commercial reduction
  for (iyear=styr_cR_agec;iyear<=endyr_cR_agec;iyear++)
  {
    ErrorFree_cR_agec(iyear)=L_cR_num_agec(iyear)/
                          sum(L_cR_num_agec(iyear));
    pred_cR_agec(iyear)=age_error*ErrorFree_cR_agec(iyear);
  }
  //cout << "FINISHED" << endl;

////-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
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*wgt_cR_mt(endyr);

FUNCTION get_msy

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

    F_L_age_msy=F_msy(ff)*sel_wgted_L;
    Z_age_msy=wgted_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,wgted_reprod));


    //Compute equilibrium values of R (including bias correction), SSB and Yield at each F
    if(set_SR_switch>1) //Beverton-Holt
    {
      R_eq(ff)=(R0/((5.0*steep-1.0)*spr_msy(ff)))*
                 (BiasCor*4.0*steep*spr_msy(ff)-wgted_spr_F0*(1.0-steep));
    }
    if(set_SR_switch<2) //Ricker
    {
      R_eq(ff)=R0/spr_msy(ff)*(1+log(BiasCor*spr_msy(ff))/steep);
    }
    if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
    N_age_msy*=R_eq(ff);
    N_age_msy_mdyr*=R_eq(ff);

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


    SSB_eq(ff)=sum(elem_prod(N_age_msy_mdyr,wgted_reprod));
    B_eq(ff)=sum(elem_prod(N_age_msy,wgt_spawn_mt(endyr)));
    L_eq_mt(ff)=sum(elem_prod(L_age_msy,wgt_wgted_L_mt));
    L_eq_knum(ff)=sum(L_age_msy);
  }

  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

  sigma_rec_dev=sqrt(var_rec_dev+dzero); //pow(var_rec_dev,0.5);  //sample SD of predicted residuals (may not equal rec_sigma)

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

  L_total_num=L_cR_num; //catch in number fish
  L_total_mt=L_cR_mt;   //landings in klb whole weight

  for(iyear=styr; iyear<=endyr; iyear++)
  {
        L_total_mt_yr(iyear)=sum(L_total_mt(iyear));
```

```
        L_total_knum_yr(iyear)=sum(L_total_num(iyear));

        B(iyear)=elem_prod(N(iyear),wgt_spawn_mt(iyear));
        totN(iyear)=sum(N(iyear));
        totB(iyear)=sum(B(iyear));
  }
  //B(endyr+1)=elem_prod(N(endyr+1),wgt_spawn_mt(endyr));
  //totN(endyr+1)=sum(N(endyr+1));
  //totB(endyr+1)=sum(B(endyr+1));

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

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

   //fill in log recruitment deviations for yrs they are nonzero
   for(iyear=styr_rec_dev; iyear<=endyr; iyear++)
   {
    log_rec_dev_output(iyear)=log_rec_dev(iyear);
   }
   //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);
   }

   //Compute the exploitation rate for ages 1+ and pop wgtd F for ages 2+
    for(iyear=styr; iyear<=endyr; iyear++)
    {
      E(iyear)=sum(L_cR_num(iyear)(2,nages))/sum(N(iyear)(2,nages));
      F_age2plus(iyear)=((F_cR(iyear)(3,nages))*N(iyear)(3,nages))/sum(N(iyear)(3,nages));
      F_cR_age2plus(iyear)=(F_cR(iyear)(3,nages)*N(iyear)(3,nages))/sum(N(iyear)(3,nages));
    }


//----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
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(iyear)))/spr_F0(iyear);
  }


  cout << "sel_wgted_L = " << sel_wgted_L  << endl;
  cout << "wgted_M     = " << wgted_M  << endl;
  cout << "wgted_reprod = "  << wgted_reprod << endl;
  cout << "wgt_wgted_L_mt = " << wgt_wgted_L_mt << endl;

  //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=wgted_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)));
    F_spr_age2plus(ff)=F_L_age_spr(3,nages)*N_age_spr(3,nages)/sum(N_age_spr(3,nages));
    spr_spr(ff)=sum(elem_prod(N_age_spr,wgted_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); //in mt
    }
  }
```

```
FUNCTION get_effective_sample_sizes

      neff_cR_agec_allyr_out=missing;
      neff_gill_lenc_allyr_out=missing;

      for (iyear=styr_cR_agec; iyear<=endyr_cR_agec; iyear++)
         {if (nsamp_cR_agec(iyear)>=minSS_cR_agec)
            { numer=sum( elem_prod(pred_cR_agec(iyear),(1.0-pred_cR_agec(iyear))) );
              denom=sum( square(obs_cR_agec(iyear)-pred_cR_agec(iyear)) );
                if (denom>0.0) {neff_cR_agec_allyr_out(iyear)=numer/denom;}
                else {neff_cR_agec_allyr_out(iyear)=-missing;}
            } else {neff_cR_agec_allyr_out(iyear)=-99;}
         }

      for (iyear=styr_gill_lenc; iyear<=endyr_gill_lenc; iyear++)
         {if (nsamp_gill_lenc(iyear)>=minSS_gill_lenc)
            { numer1=sum( elem_prod(pred_gill_lenc(iyear),(1.0-pred_gill_lenc(iyear))) );
              denom1=sum( square(obs_gill_lenc(iyear)-pred_gill_lenc(iyear)) );
                if (denom1>0.0) {neff_gill_lenc_allyr_out(iyear)=numer1/denom1;}
                else {neff_gill_lenc_allyr_out(iyear)=-missing;}
            } else {neff_gill_lenc_allyr_out(iyear)=-99;}
         }

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

FUNCTION get_Fmed_benchmarks

  //sorting function for recruitment and SPR values (slow algorithm, but works)
  R_temp=rec(styr_bench,endyr_bench);
  SPR_temp=pred_SPR(styr_bench,endyr_bench);
  for(int jyear=endyr_bench; jyear>=styr_bench; jyear--)
  {
    R_sort(jyear)=max(R_temp);
    SPR_sort(jyear)=max(SPR_temp);
    for(iyear=styr_bench; iyear<=endyr_bench; 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_bench-styr_bench)*0.5;
  quant_whole=(endyr_bench-styr_bench)*0.5;
  quant_diff=quant_decimal-quant_whole;
  R_med=R_sort(styr_bench+quant_whole)*(1-quant_diff)+R_sort(styr_bench+quant_whole+1)*(quant_diff);
  SPR_med=SPR_sort(styr_bench+quant_whole)*(1-quant_diff)+SPR_sort(styr_bench+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_bench-styr_bench)*0.75;
  quant_whole=(endyr_bench-styr_bench)*0.75;
  quant_diff=quant_decimal-quant_whole;
  SPR_75th=SPR_sort(styr_bench+quant_whole)*(1-quant_diff)+SPR_sort(styr_bench+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);
    }
  }

FUNCTION evaluate_objective_function
  fval=0.0;
  fval_unwgt=0.0;
```

```
////---likelihoods---------------------------

////---Indices-------------------------------
  f_JAIs_cpue=0.0;
  f_JAIs_cpue=lk_lognormal(pred_JAIs_cpue,obs_JAIs_cpue_final,JAIs_cpue_cv_final,w_I_JAIs);
  fval+=f_JAIs_cpue;
  fval_unwgt+=f_JAIs_cpue;

  //f_JAIt_cpue=0.0;
  //f_JAIt_cpue=lk_lognormal(pred_JAIt_cpue,obs_JAIt_cpue_final,JAIt_cpue_cv_final,w_I_JAIt);
  //fval+=f_JAIt_cpue;
  //fval_unwgt+=f_JAIt_cpue;

  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_unwgt+=f_gill_cpue;

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

  f_cR_L=0.0; //in 1000 mt
  f_cR_L=lk_lognormal(pred_cR_L_mt(styr,endyr),obs_cR_L(styr,endyr),
                      cR_L_cv(styr,endyr),w_L);
  fval+=f_cR_L;
  fval_unwgt+=f_cR_L;



//////---Age comps-------------------------------
  //f_cR_agec=100.0;
  //f_cR_agec=lk_multinomial(nsamp_cR_agec,pred_cR_agec,obs_cR_agec,nyr_cR_agec, minSS_cR_agec, w_ac);
  //fval+=f_cR_agec;
  //fval_unwgt+=f_cR_agec;

  f_cR_agec=0.0;
  for (iyear=styr_cR_agec; iyear<=endyr_cR_agec; iyear++)
  {
    if (nsamp_cR_agec(iyear)>=minSS_cR_agec)
    {
    f_cR_agec-=neff_cR_agec(iyear)*
            sum(elem_prod((obs_cR_agec(iyear)+dzero),
               log(elem_div((pred_cR_agec(iyear)+dzero),
                  (obs_cR_agec(iyear)+dzero)))));
    }
  }

  fval+=w_ac*f_cR_agec;
  fval_unwgt+=f_cR_agec;



////---Length comps-------------------------------
  f_gill_lenc=0.0;

  //cout << "nsamp_gill_lenc" << nsamp_gill_lenc << endl;
  //cout << "pred_gill_lenc" << pred_gill_lenc << endl;
  //cout << "obs_gill_lenc" << obs_gill_lenc << endl;
  //cout << "nyr_gill_lenc" << nyr_gill_lenc << endl;
  //cout << "minSS_gill_lenc" << minSS_gill_lenc << endl;
  //cout << "weight" << w_I_gill_lc << endl;

  //f_gill_lenc=lk_multinomial(nsamp_gill_lenc,pred_gill_lenc,obs_gill_lenc,nyr_gill_lenc,minSS_gill_lenc,w_I_gill_lc);
  //cout << "gill_lenc_like" << f_gill_lenc << endl;

  //fval+=f_gill_lenc;
 // fval_unwgt+=f_gill_lenc;

  for (iyear=styr_gill_lenc; iyear<=endyr_gill_lenc; iyear++)
  {
    if (nsamp_gill_lenc(iyear)>=minSS_gill_lenc)
    {
    f_gill_lenc-=neff_gill_lenc(iyear)*
        sum(elem_prod((obs_gill_lenc(iyear)+dzero),
            log(elem_div((pred_gill_lenc(iyear)+dzero),
               (obs_gill_lenc(iyear)+dzero)))));
    }
  }

  fval+=w_I_gill_lc*f_gill_lenc;
  fval_unwgt+=f_gill_lenc;


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

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

  //f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
  //if (styr_rec_dev<endyr_rec_phase1)
  //  {
  //    f_rec_dev_early=pow(log_rec_dev(styr_rec_dev),2);
```

```
 //    for(iyear=(styr_rec_dev+1); iyear<=endyr_rec_phase1; iyear++)
 //    {f_rec_dev_early+=pow(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1),2);}
 //  }
 //fval+=w_rec_early*f_rec_dev_early;

 //f_rec_dev_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 (endyr_rec_phase2<endyr)
 // {
 //    for(iyear=(endyr_rec_phase2+1); iyear<=endyr; iyear++)
 //    {f_rec_dev_end+=pow(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1),2);}
 //  }
 //fval+=w_rec_end*f_rec_dev_end;

 //f_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;
  //}

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

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

 // fval+=w_fullF*f_fullF_constraint;

 //Random walk components of fishery dependent indices
// f_PN_RW_cpue=0.0;
// for (iyear=styr_PN_cpue; iyear<endyr_PN_cpue; iyear++)
//      {f_PN_RW_cpue+=square(q_RW_log_dev_PN(iyear))/(2.0*set_q_RW_PN_var);}
// fval+=f_PN_RW_cpue;

 //JAI combination weights penalty to sum to 1.0
 //f_JAI_wgts=0.0;
 //f_JAI_wgts=square(1.0-(wgt_JAI1+wgt_JAI2+wgt_JAI3+wgt_JAI4));
 //fval+=w_JAI_wgts*f_JAI_wgts;

 f_priors=0.0;
 f_priors=norm2(log_Nage_dev);
 //f_priors+=neg_log_prior(steep,set_steep,square(set_steep_se),4);
 //f_priors+=square(R_autocorr-set_R_autocorr);
 //f_priors+=square(q_DD_beta-set_q_DD_beta)/square(set_q_DD_beta_se);
 //f_priors+=neg_log_prior(Linf,set_Linf,square(set_Linf_se),3);
 //f_priors+=neg_log_prior(K,set_K,square(set_K_se),3);
 //f_priors+=neg_log_prior(t0,set_t0,square(set_t0_se),3);
 //f_priors+=neg_log_prior(rec_sigma,set_rec_sigma,square(set_rec_sigma_se),3);


 //f_priors+=sum(square(len_cv-set_len_cv));
 //f_priors+=neg_log_prior(len_cv(1),set_len_cv(1),square(set_len_cv_se(1)),3);
 //f_priors+=neg_log_prior(len_cv(2),set_len_cv(2),square(set_len_cv_se(2)),3);
 //f_priors+=neg_log_prior(len_cv(3),set_len_cv(3),square(set_len_cv_se(3)),3);
 //f_priors+=neg_log_prior(len_cv(4),set_len_cv(4),square(set_len_cv_se(4)),3);
 //f_priors+=neg_log_prior(len_cv(5),set_len_cv(5),square(set_len_cv_se(5)),3);

 //f_priors+=neg_log_prior(selpar_L50_gill, set_selpar_L50_gill, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope_gill, set_selpar_slope_gill, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_L502_gill, set_selpar_L502_gill, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope2_gill, set_selpar_slope2_gill, -1.0, 3);

 //f_priors+=neg_log_prior(sel_age0_gill_logit, set_sel_age0_gill, -1.0, 3);
 f_priors+=neg_log_prior(sel_age1_gill_logit, set_sel_age1_gill, -1.0, 3);
 //f_priors+=neg_log_prior(sel_age2_gill_logit, set_sel_age2_gill, -1.0, 3);
 f_priors+=neg_log_prior(sel_age3_gill_logit, set_sel_age3_gill, -1.0, 3);
 f_priors+=neg_log_prior(sel_age4_gill_logit, set_sel_age4_gill, -1.0, 3);

 //f_priors+=neg_log_prior(selpar_L50_cR1, set_selpar_L50_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope_cR1, set_selpar_slope_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_L50_cR3, set_selpar_L50_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope_cR3, set_selpar_slope_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_L50_cR4, set_selpar_L50_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope_cR4, set_selpar_slope_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_L502_cR1, set_selpar_L502_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope2_cR1, set_selpar_slope2_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_L502_cR4, set_selpar_L502_cR, -1.0, 3);
 //f_priors+=neg_log_prior(selpar_slope2_cR4, set_selpar_slope2_cR, -1.0, 3);
```

```
  //f_priors+=neg_log_prior(sel_age0_cR1_logit, set_sel_age0_cR1, -1.0, 3);
  for (iyear=styr;iyear<=endyr_period2;iyear++)
    {
     f_priors+=neg_log_prior(sel_age1_cR1_logit(iyear), set_sel_age1_cR1, -1.0, 3);
     //f_priors+=neg_log_prior(sel_age2_cR1_logit, set_sel_age2_cR1, -1.0, 3);
     f_priors+=neg_log_prior(sel_age3_cR1_logit(iyear), set_sel_age3_cR1, -1.0, 3);
     f_priors+=neg_log_prior(sel_age4_cR1_logit(iyear), set_sel_age4_cR1, -1.0, 3);
    }

  //f_priors+=neg_log_prior(sel_age0_cR3_logit, set_sel_age0_cR3, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age1_cR3_logit, set_sel_age1_cR3, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age2_cR3_logit, set_sel_age2_cR3, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age3_cR3_logit, set_sel_age3_cR3, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age4_cR3_logit, set_sel_age4_cR3, -1.0, 3);

  //f_priors+=neg_log_prior(sel_age0_cR4_logit, set_sel_age0_cR4, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age1_cR4_logit, set_sel_age1_cR4, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age2_cR4_logit, set_sel_age2_cR4, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age3_cR4_logit, set_sel_age3_cR4, -1.0, 3);
  //f_priors+=neg_log_prior(sel_age4_cR4_logit, set_sel_age4_cR4, -1.0, 3);


  if(switch_prior==1)
  {
    fval+=f_priors;
  }
//  cout << "fval = " << fval << "  fval_unwgt = " << fval_unwgt << 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;
```

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

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

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

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

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

  //><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
  report << "Likelihood " << "Value " << "Weight" << endl;
  report << "JAI_seine_index " << f_JAIs_cpue << " " << w_I_JAIs << endl;
  //report << "JAI_trawl_index " << f_JAIt_cpue << " " << w_I_JAIt << endl;
  report << "Gillnet_index " << f_gill_cpue << " " << w_I_gill << endl;
  report << "Gillnet_lenc " << f_gill_lenc << " " << w_I_gill_lc << endl;
```

```
  report << "reduction_agec " << f_cR_agec << " " << w_ac << endl;
  report << "L_reduction " << f_cR_L << " " << w_L << endl;
  report << "R_dev " << f_rec_dev << " " << w_rec << endl;
  //report << "R_dev_early " << f_rec_dev_early << " " << w_rec_early << endl;
  //report << "R_dev_end " << f_rec_dev_end << " " << w_rec_end << endl;
  //report << "F_tune " << f_Ftune << " " << w_Ftune << endl;
  //report << "fullF_constraint " << f_fullF_constraint << " " << w_fullF << endl;
  report << "priors " << f_priors << " " << switch_prior << endl;

  report << "TotalLikelihood " << fval << endl;
  report << "UnwgtLikelihood " << fval_unwgt << endl;

  report << "Error levels in model" << endl;
  report << "JAI_seine_cv " << JAIs_cpue_cv << endl;
  report << "JAI_trawl_cv " << JAIt_cpue_cv << endl;
  report << "Gillnet_cv " << gill_cpue_cv << endl;
  report << "L_reduction_cv " << cR_L_cv << endl;

  report << "NaturalMortality Vector" << endl;
  report << "Age " << agebins << endl;
  report << "M_vector " << M << endl;
  report << "NaturalMortality Matrix " << endl;
  report << "Year " << agebins << endl;
  for(iyear=styr; iyear<=endyr; iyear++)
  {
    report << iyear << " " << M_mat(iyear) << endl;
  }

  report << "Steepness " << steep << endl;
  report << "R0 " << R0 << endl;

  report << "Recruits" << endl;
  report << "Year";
  for(iyear=styr; iyear<=endyr; iyear++)
  {
    report << " " << iyear;
  }
  report << endl;
  report << "Age-0_recruits " << column(N,1) << endl;
  report << "Age-1_recruits " << column(N,2) << endl;
  report << "SSB" << endl;
  report << "Year";
  for(iyear=styr; iyear<=endyr; iyear++)
  {
    report << " " << iyear;
  }
  report << endl;
  report << "FEC " << SSB << endl;
  //report << "SSB " << FEC << endl;
  report << "Lagged_R " << column(N,1)(styr+1,endyr) << endl;
  report << "wgt_wgted_L_mt" << wgt_wgted_L_mt << endl;

  report << "nsamp_cR_agec_allyr" << nsamp_cR_agec_allyr << endl;

//    cout<< mfexp(log_len_cv)<<endl;
//  report << "TotalLikelihood " << fval << endl;
  #include "gmenhad_make_Robject003.cxx"   // write the S-compatible report
  }
```

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

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##  Data Input File
##  GSMFC Assessment: Gulf Menhaden
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#starting and ending year of model
1948
2010
#Starting year to estimate recruitment deviation from S-R curve
1948
#3 phases of constraints on recruitment deviations: allows possible heavier constraint in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1967
2008
#4 periods of changing selectivity for reduction fishery: yr1-1970, 1971-1979, 1980-1993, 1994-2010---right now only using 2 periods yr1-1993 and 1994-2010
#ending years of regulation period
1963
1979
1993

#2 periods of changing selectivity for gillnet survey:  yr1-1993, 1994-2010
#ending year of early period
1993

#starting and ending years to use for benchmark calculations
1948
2010

#Number of ages (last age is plus group)
5
##vector of agebins, last is a plus group
0 1 2 3 4

#number length bins used to match length comps and number used to compute plus group
#26
#5
51
10

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
#10 30 50 70 90 110 130 150 170 190 210 230 250 270 290 310 330 350 370 390 410
#430 450 470 490 510
#10 30 50 70 90 110 130 150 170 190 210 230 250 270 290 310 330 350 370 390 410 430 450 470 490 510
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145 155 165 175 185 195 205 215 225 235 245 255 265 275 285 295 305 315 325 335 345 355 365 375 385 395 405     415 425 435 445 455 465 475 485 495
505
415 425 435 445 455 465 475 485 495 505
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145 155 165 175 185 195 205 215 225 235 245 255 265 275 285 295 305 315 325 335 345 355 365 375 385 395 405

#max value of F used in spr and msy calculations
10.0
#number of iterations in spr calculations
30001
#number of iterations in msy calculations
30001
#Number years at end of time series over which to average sector Fs, for weighted selectivities
47
#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

##time-invariant vector of % maturity-at-age for females (ages 0-6+)
0.0 0.0 1       1 1
#0.0      0.2     1       1       1         #for a sensitivity run
##time-invariant vector of % maturity-at-age for males (ages 0-6+)
0.0 0.0 1 1 1
#0.0      0.2     1       1       1              #for a sensitivity run
#time-invariant vector of proportion female (ages 0-6+)
0.5 0.5 0.5 0.5 0.5
#time of year (as fraction) for spawning: Jan 1=0d/365d
0.0
#age-dependent natural mortality at age
1.67 1.31 1.1 1.0 0.94   #scaled to tagging data
#1.38 1.09 0.91 0.82 0.78 0.75 0.73  #scaled to lower tagging for sensitivity run
#1.93 1.52 1.27 1.15 1.09 1.05 1.02  #scaled to upper tagging for sensitivity run
#1.07 0.84 0.7 0.64 0.6 0.58 0.57  #scaled to Hoenig estimate
#age-independent natural mortality (used only to compute MSST=(1-M)SSBmsy)
1.1
#age and year specific natural mortality
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
```

```
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94
1.67 1.31 1.10 1.00 0.94


##Spawner-recruit parameters
#switch for S-R function to use Ricker (1) or Beverton-Holt (2)
2
#steepness (fixed or initial guess)
0.9
#standard error of steepness (from meta-analysis)
0.2
#log_R0 - log virgin recruitment
2.7
# R autocorrelation
0.0
# SD of recruitment in log space
0.5
# SE of SD recruitment
0.25


##--><>--><>--><>-- Weight-at-age in the fishery (g) --><>--><>--><>--><>--><>--><>--><>--><>
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
32.8 67.8 125.9 189.9 254.8
33.3 71.4 126.9 176.1 214.6
30.6 62.4 124.1 205.8 303.9
34.6 69.6 126.8 187.8 245.9
26.0 65.8 122.8 169.5 202.9
29.8 69.3 135.9 205.6 269.5
38.4 68.9 123.2 190.8 268.7
28.5 72.0 133.0 181.5 215.0
30.6 73.1 135.2 188.4 228.0
30.5 79.6 139.7 179.9 203.2
42.7 83.4 154.1 236.5 323.3
26.3 86.9 157.7 199.2 220.0
40.0 76.6 146.7 240.4 355.5
31.8 69.3 134.7 208.8 283.8
27.6 57.3 112.1 179.8 255.4
39.7 69.9 123.7 190.6 267.8
37.1 73.1 124.9 171.5 209.1
12.6 56.0 130.2 191.4 232.2
23.8 59.5 114.6 164.6 203.9
37.2 67.5 116.9 170.9 224.7
26.7 68.0 128.2 178.3 214.1
```

```
20.2 62.0 125.5 178.0 214.5
21.5 60.6 118.2 165.0 197.7
16.3 52.4 112.2 166.3 207.6
30.3 58.7 103.7 150.0 192.9
20.0 57.9 112.9 156.8 186.7
29.4 62.3 113.5 163.5 206.8
26.9 66.9 125.6 175.6 212.5
42.1 79.4 132.1 179.6 218.0
40.0 78.5 129.5 171.7 203.1
43.6 80.5 132.7 180.1 219.0
32.3 71.1 125.3 171.1 205.2
39.6 79.3 135.4 184.6 223.1
31.7 68.1 123.2 174.4 216.3
27.4 69.6 126.7 170.1 198.7
35.4 73.1 123.7 165.1 195.3
49.5 84.5 132.5 175.5 210.9
35.3 68.8 112.0 146.3 171.0
65.3 97.7 140.7 179.6 212.6
34.8 74.2 125.4 165.0 192.1
27.8 61.3 110.3 153.7 187.7
30.9 64.5 111.2 150.9 180.7
46.6 73.4 110.7 146.3 177.8
29.8 65.5 109.9 142.0 162.7
33.4 69.5 116.8 154.0 180.0
36.5 78.7 127.9 160.8 180.3
57.8 83.8 119.1 152.6 182.5
26.3 69.2 120.8 154.3 173.3


##--><>--><>--><>-- Weight-at-age - start of year (g) --><>--><>--><>--><>
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 43.3 95.7 157.6 222.4
0.00 45.0 99.3 152.7 196.7
0.00 39.8 90.5 162.7 253.1
0.00 45.2 97.2 157.3 217.5
0.00 38.1 94.9 147.8 187.8
0.00 41.4 101.4 171.0 238.6
0.00 47.6 94.2 155.5 228.7
0.00 41.8 103.4 159.2 200.0
0.00 43.6 104.6 163.4 209.9
0.00 46.0 111.8 162.3 193.3
0.00 54.9 116.8 194.3 279.7
0.00 45.1 125.9 181.8 211.5
0.00 50.7 108.6 190.7 295.4
0.00 42.9 100.3 171.2 246.6
0.00 36.3 82.7 144.6 216.9
0.00 48.8 95.0 155.7 228.1
0.00 48.4 99.2 149.2 191.5
0.00 24.2 93.3 163.3 214.2
0.00 34.4 87.0 140.8 185.7
0.00 46.5 91.2 143.6 198.1
0.00 39.2 98.7 155.0 197.9
0.00 32.4 94.2 153.7 198.1
0.00 33.1 90.0 143.4 183.0
0.00 26.6 82.0 140.7 188.7
0.00 39.0 80.6 127.0 172.0
0.00 31.3 86.1 136.7 173.4
0.00 39.4 87.5 139.1 186.2
0.00 39.0 96.6 152.1 195.6
0.00 53.9 106.0 156.9 200.0
0.00 52.3 104.7 152.0 188.7
0.00 55.3 106.8 157.3 200.6
0.00 44.4 98.8 149.6 189.6
0.00 52.0 107.7 161.3 205.2
0.00 42.9 95.5 149.7 196.6
0.00 40.5 99.3 150.4 186.1
0.00 47.4 99.2 145.8 181.6
0.00 60.7 108.8 154.9 194.2
0.00 46.1 91.2 130.4 159.8
0.00 75.9 119.5 160.8 196.9
0.00 47.3 100.9 146.8 179.9
0.00 38.1 86.0 133.0 171.9
0.00 41.4 88.3 132.2 167.0
0.00 55.3 92.1 128.9 162.6
0.00 41.3 89.0 127.6 153.6
0.00 44.9 94.1 136.9 168.3
0.00 50.3 105.2 146.4 171.9
0.00 66.3 101.5 136.2 168.0
0.00 39.9 97.1 139.8 165.3


##--><>--><>--><>-- Fecundity-at-age - not adjusted for maturity (number of maturing ova per individual) --><>--><>--><>--><>
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
```

```
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 8493 21641 38974 58568
0.0 9310 23161 38020 50894
0.0 7720 20105 39791 66601
0.0 8448 21657 39110 58210
0.0 7075 22273 38862 52492
0.0 7855 23500 44562 66991
0.0 9294 21812 40772 65957
0.0 8074 24020 40373 53114
0.0 8615 24348 41351 55631
0.0 9188 27162 42783 52917
0.0 10122 25417 47259 73668
0.0 8302 31388 50499 61438
0.0 10185 26047 52154 89510
0.0 8135 24503 49053 78743
0.0 7080 19862 39982 66398
0.0 9954 22119 39955 63125
0.0 10391 24331 39507 53133
0.0 4816 22174 41800 56802
0.0 6401 19909 35899 50373
0.0 9570 20845 35182 50972
0.0 7692 22931 39107 52201
0.0 5874 21289 38409 52161
0.0 6243 21518 38301 51818
0.0 4489 17635 33953 48483
0.0 7392 17956 31283 45328
0.0 5324 18838 33512 45084
0.0 7230 18982 33275 47351
0.0 6904 20835 36215 49185
0.0 10585 23594 37548 50055
0.0 9136 22873 37426 49828
0.0 10052 22850 37031 50148
0.0 8296 23083 39286 53176
0.0 10001 23829 38581 51452
0.0 8830 22398 37748 51799
0.0 7601 22869 38026 49370
0.0 8730 22434 36724 48589
0.0 12299 25058 38570 50845
0.0 8846 22503 36686 48445
0.0 15164 27114 39653 51381
0.0 9419 24290 38868 50145
0.0 7201 20208 35159 48665
0.0 7877 19700 32115 42592
0.0 10773 20634 31671 42563
0.0 7744 20214 31712 39984
0.0 8597 21106 33281 42765
0.0 10110 24183 35707 43176
0.0 14042 24338 35553 46606
0.0 7259 22489 35755 44264


##-->&lt;>-->&lt;>-->&lt;>-- Juvenile Abundance Index from seine surveys -->&lt;>-->&lt;>-->&lt;>-->&lt;>-->&lt;>
##Switch to use single index (=1) or let model combine indices (not equal to 1)
1
##Starting and ending years of time series, respectively
1977
2010
##Observed CPUE (numbers) and CV vectors, respectively
0.8532117 0.9539007 0.3765042 0.7936619 0.6447944 1.1169395 0.7439912 2.6041844 0.766218 2.0378689 0.7151532 0.5974039 0.6396402 1.1625989 1.1488432 1.2745629 1.6400419 0.6244874 0.8701708 1.4388904
0.8811541 1.3375117 0.871577 0.4719151 1.012841 0.8708502 1.0712156 0.6170468 0.7083236 0.6457001 0.9499249 0.5974745 1.0092283 1.9521695
0.48118822 0.389104138 0.342169217 0.32661541 0.237447979 0.200948356 0.186093321 0.159674922 0.196481516 0.140792789 0.149082264 0.15762195 0.144771109 0.119772413 0.120532936 0.116682878 0.114211781
0.1250578 0.116892518 0.112876121 0.109112583 0.107422125 0.117222422 0.125026892 0.109825483 0.107561704 0.107970075 0.109646646 0.107792881 0.11475117 0.102703325 0.106362308 0.106086634 0.108682567


##-->&lt;>-->&lt;>-->&lt;>-- Juvenile Abundance Indices (4 groups) from seine surveys -->&lt;>-->&lt;>-->&lt;>-->&lt;>-->&lt;>
##Series 1 Observed CPUE (numbers) and CV vectors, respectively
##must have zeros in place of missing values and all series must be the same length as single index above
0.8532117 0.9539007 0.3765042 0.7936619 0.6447944 1.1169395 0.7439912 2.6041844 0.766218 2.0378689 0.7151532 0.5974039 0.6396402 1.1625989 1.1488432 1.2745629 1.6400419 0.6244874 0.8701708 1.4388904
0.8811541 1.3375117 0.871577 0.4719151 1.012841 0.8708502 1.0712156 0.6170468 0.7083236 0.6457001 0.9499249 0.5974745 1.0092283 1.9521695
0.48118822 0.389104138 0.342169217 0.32661541 0.237447979 0.200948356 0.186093321 0.159674922 0.196481516 0.140792789 0.149082264 0.15762195 0.144771109 0.119772413 0.120532936 0.116682878 0.114211781
0.1250578 0.116892518 0.112876121 0.109112583 0.107422125 0.117222422 0.125026892 0.109825483 0.107561704 0.107970075 0.109646646 0.107792881 0.11475117 0.102703325 0.106362308 0.106086634 0.108682567
##Series 2 Observed CPUE (numbers) and CV vectors, respectively
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##Series 3 Observed CPUE (numbers) and CV vectors, respectively
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##Series 4 Observed CPUE (numbers) and CV vectors, respectively
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

##-->&lt;>-->&lt;>-->&lt;>-- Juvenile Abundance Index from trawl surveys -->&lt;>-->&lt;>-->&lt;>-->&lt;>-->&lt;>
##Starting and ending years of time series, respectively
1967
2010
##Observed CPUE (numbers) and CV vectors, respectively
0.5726388 2.6901819 1.2409177 1.117585 0.5862024 0.3520756 0.6618439 1.0078254 1.5242414 1.6389502 2.1070125 1.0843953 0.632725 1.0377783 0.7644635 0.7962733 0.9315196 1.4954697 0.7443739 0.7183562
0.749796 0.6869988 0.5638866 0.8771056 0.8758685 1.1675068 1.1723505 0.8211532 0.6677382 0.9882219 0.732906 1.0012529 0.6989131 0.5152103 1.053807 0.9527749 0.9331622 0.7879501 0.7512019 0.6815322
1.1871579 0.6214085 0.7289429 3.0783241
0.16 0.16 0.21 0.15 0.18 0.16 0.13 0.09 0.16 0.20 0.14 0.12 0.13 0.17 0.11 0.07 0.08 0.08 0.07 0.08 0.07 0.07 0.08 0.08 0.07 0.06 0.07 0.08 0.07 0.07 0.06 0.07 0.08 0.07 0.07 0.06 0.06 0.06 0.07
0.07 0.07 0.07 0.07 0.08

##-->&lt;>-->&lt;>-->&lt;>-- Adult Abundance Index from gillnet surveys -->&lt;>-->&lt;>-->&lt;>-->&lt;>-->&lt;>
##Starting and ending years of time series, respectively
1986
2010
##Observed CPUE (numbers) and CV vectors, respectively
```

```
0.8876313 0.5516841 1.0535439 0.7457594 0.7682163 0.7921094 0.6020293 0.5489181 0.8391153 0.6947594 0.7646888 1.1077033 1.0512145 0.8004876 1.0328651 1.1544858 1.0518452 1.0397885 0.9415808 1.1624293
1.1342473 0.9203316 2.4410209 2.0918482 0.8216967
0.08336033 0.09139956 0.07209881 0.07521872 0.07851684 0.08009715 0.08651338 0.08949987 0.0851033 0.0906863 0.08141417 0.07904502 0.07571395 0.07672932 0.07134729 0.07843334 0.07403435 0.06792069
0.0751475 0.07092119 0.06705314 0.06876212 0.06675556 0.0606661 0.07674898

#Number of years, start year, end year,  and vector of years of length compositions for gillnet survey
25
1986
2010
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
#sample size of gillnet survey length comp data by year (first row observed N, second row effective N: effective may be set to observed)
#351 235 35 4 4 5 82 196 194 213 262 280 366 325 410 353 383 387 348 374 460 375 439 461 287
#351 235 35 4 4 5 82 196 194 213 262 280 366 325 410 353 383 387 348 374 460 375 439 461 287
200     200     35      4       4       5       82      196     194     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200
200     200     35      4       4       5       82      196     194     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200

#length composition samples (year,lengthbin 10 mm)
#unweighted length comps
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.02 0.03 0.10 0.23 0.24 0.12 0.10 0.05 0.02 0.02 0.02 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.02 0.01 0.02 0.07 0.22 0.28 0.11 0.06 0.03 0.03 0.03 0.05 0.03 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.03 0.06 0.16 0.10 0.12 0.24 0.12 0.06 0.07 0.01 0.00 0.00 0.00 0.00 0.01 0.00 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.20 0.20 0.00 0.00 0.20 0.20 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.20 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.08 0.15 0.00 0.08 0.08 0.00 0.15 0.00 0.15 0.00 0.15 0.08 0.00 0.08 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.11 0.22 0.22 0.33 0.11 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.03 0.06 0.07 0.08 0.16 0.19 0.12 0.08 0.07 0.06 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.03 0.10 0.13 0.08 0.14 0.18 0.10 0.06 0.06 0.05 0.03 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.05 0.14 0.16 0.08 0.09 0.12 0.09 0.08 0.07 0.06 0.03 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.06 0.16 0.14 0.09 0.10 0.13 0.09 0.06 0.05 0.04 0.03 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.02 0.06 0.15 0.16 0.09 0.11 0.11 0.08 0.05 0.05 0.04 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.01 0.05 0.12 0.14 0.11 0.11 0.14 0.10 0.07 0.06 0.03 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.07 0.17 0.17 0.11 0.10 0.12 0.08 0.04 0.04 0.02 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.08 0.16 0.14 0.09 0.10 0.12 0.09 0.07 0.06 0.03 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.02 0.07 0.14 0.11 0.07 0.07 0.12 0.12 0.09 0.08 0.06 0.03 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.05 0.11 0.10 0.08 0.11 0.13 0.10 0.08 0.08 0.06 0.03 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.02 0.03 0.06 0.13 0.14 0.09 0.09 0.10 0.08 0.06 0.07 0.05 0.04 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.09 0.18 0.18 0.10 0.10 0.13 0.08 0.03 0.03 0.03 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.02 0.04 0.11 0.17 0.17 0.10 0.10 0.09 0.07 0.04 0.04 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.07 0.15 0.17 0.11 0.11 0.13 0.09 0.04 0.04 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.02 0.09 0.16 0.15 0.14 0.10 0.11 0.08 0.05 0.03 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.02 0.05 0.14 0.17 0.14 0.14 0.13 0.07 0.04 0.03 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.02 0.04 0.13 0.15 0.09 0.11 0.15 0.11 0.07 0.06 0.03 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.02 0.04 0.09 0.13 0.10 0.11 0.12 0.12 0.08 0.08 0.05 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.02 0.07 0.14 0.12 0.08 0.10 0.11 0.10 0.07 0.06 0.05 0.02 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00


##-->><>-->><>-->><>-- Commercial Reduction fishery -->><>-->><>-->><>-->><>-->><>-->><>-->><>
#Starting and ending years of landings time series, respectively
1948
2010
##Observed landings (1000 mt) and assumed CVs
#total landings including reduction, bait, and recretional
74.8187801 107.6187801 147.4101074 155.0130558 227.3137815 195.9112867 181.4112414 213.5209029 244.2244863 159.513464 196.4501597 326.1193153 377.0154598 456.1215379 479.2190432 437.7302923 408.0482092
461.6060142 358.0642555 316.3678044 372.3173086 521.8473646 546.3899289 729.0759754 502.4022666 487.0559892 587.9286657 543.0215724 561.7376468 447.6076643 820.6137239 779.8369297 702.5088237
553.7127019 855.5306431 925.26319 985.1222814 884.5466744 830.8792809 911.6692996 640.2078039 583.5437816 539.5201795 552.9801032 432.8065373 551.552376 774.9238254 472.0244485 491.7517232 623.1465888
487.1615644 685.3769203 580.294677 522.0985091 575.0798298 517.7067425 469.1853195 434.1293029 464.6286697 454.0805233 425.567131 457.6892615 379.9389777
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 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 reduction, bait, and recreational  fishery combined
1964
2010
47
1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 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
##sample sizes of age comps by year  (first row observed N, second row effective N: effective may be set to observed)
#number of fish sampled first row, number of net sets 2 and 3 rows
#12260 15185 12429 14065 15273 14764 10402 7654 9886 8953 10086 9527 13389 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
#625 790 640 721 795 759 527 393 998 896 1009 953 1355 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
#625 790 640 721 795 759 527 393 998 896 1009 953 1355 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
#200    200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     20
#200    200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     200     20
10 10 10 10 10 10 10 10 10 10 10 10 10 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200
10 10 10 10 10 10 10 10 10 10 10 10 10 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200


#age composition samples (year,age)
```

```
0.001 0.673 0.302 0.024 0.001
0.007 0.807 0.173 0.013 0.000
0.007 0.781 0.204 0.008 0.000
0.005 0.920 0.073 0.003 0.000
0.014 0.759 0.219 0.008 0.000
0.003 0.819 0.174 0.004 0.000
0.009 0.581 0.404 0.006 0.000
0.003 0.727 0.247 0.023 0.001
0.004 0.623 0.354 0.018 0.001
0.012 0.707 0.258 0.023 0.000
0.000 0.715 0.274 0.011 0.000
0.024 0.541 0.332 0.102 0.000
0.000 0.744 0.223 0.033 0.000
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
0.000 0.667 0.293 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 0.543 0.386 0.067 0.003
0.000 0.362 0.564 0.062 0.012
0.000 0.250 0.672 0.073 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

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

1.21 #selpar_L50_cR   ---commercial reduction fishery
3.56 #selpar_slope_cR
6.0 #selpar_L502_cR
0.0  #selpar_slope2_cR

1.2 #selpar_L50_gill  ---adult abundance index based on gillnet surveys
7.5 #selpar_slope_gill
3.2 #selpar_L502_gill
0.0 #selpar_slope2_gill

#vector of initial guesses for gillnet selectivity with a parameter estimated for each age
#-10.0  -10.0  10.0  10.0  10.0   #logit space
-10.0   0.915  9.918 10.0  10.0   #logit space

#vector of initial guesses for commercial reduction selectivity with a parameter estimated for each age
-10.0   0.0   10.0   0.0   0.0  #period 1
-10.0   0.0   10.0   10.0   10.0  #period 3
-10.0   0.0   10.0   10.0  10.0  #period 4


##################Likelihood Component Weighting#################################################################################
##Weights in objective fcn
1.0  #landings
0.25#0.742#1.0     #age comps
1.0#0.389#1.0 #JAI-seine index
0.0     #JAI-trawl index
1.0#2.0#0.300#1.0     #adult gillnet index
0.5#0.160#1.0     #length comps for gillnet index
1.0 #S-R residuals
0.0 #constraint on early recruitment deviations
0.0 #constraint on ending recruitment deviations
0.0     #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization)
0.0 #weight on tuning F (penalty not applied in final phase of optimization)
0.0     #weight for penalty to keep JAI combination weights summing to 1.0

##################################################################################################################
##log catchabilities (initial guesses)
-13     #JAI seine survey
-13     #JAI trawl survey
6      #gillnet survey

#exponent for JAI cpue index
1.0

#JAI combination weights
0.25
0.25
0.25
0.25
```

```
#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
-1
##annual positive rate of increase on all fishery dependent q 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

##log mean F (initial guesses) for commercial reduction, bait, and recreational combined
-0.2

#Initialization F as a proportion of  first few assessment years (set to 1.0 without evidence otherwise)
1.0

#Tuning F (not applied in last phase of optimization)
1.5

#Year for tuning F
2006

#threshold sample sizes (greater than or equal to) for gillnet length comps and reduction age comps
100.0
1.0

#switch to turn priors on/off (-1 = off, 1 = on)
1

################################################################################################################
#Ageing error matrix (columns are true age 0-6, rows are ages as read for age comps)
#1 0 0 0 0
#0 1 0 0 0
#0 0 1 0 0
#0 0 0 1 0
#0 0 0 0 1

#scale to otolith comparison
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


#################################################################################################################
#Environmental factors
###########Total River flow###################
10983.0
18437.0
16349.2
13215.0
21193.0
22515.0
17535.6
22496.0
20899.0
35071.2
35775.6
28075.8
21406.4
12878.6
22944.0
27794.4
21521.8
10943.6
21331.8
31445.0
25676.6
31048.6
27107.4
28229.2
24416.4
26665.2
24476.4
31715.2
24407.8
29912.8
30620.6
21659.4
18156.6
34671.2
25102.0
26949.2
11735.4
21751.0
23679.6
22235.8
23895.0
33908.4
14050.4
23438.2
22618.6
19011.8
33699.4

#switch for incorporation of environmental factor or not (1=on and 2=off)
```

```
2
#parameter for the environmental factor
0.005  #initial guess

##################################################################################################################
#Length at age used for gillnet survey length comps but based on reduction fishery lengths
#observed lengths at midyear
110.34 148.92 178.2 199.38 208.95

#estimated variation in growth across ages, assumed constant across time
0.126077397 0.098063335 0.063808731 0.051807243 0.049427251
#se of the length at age
0.5 0.027525088 0.026459695 0.072955378 0.264434554

#Von B intial guesses for parameters
237.8
0.444
-0.808

#Standard errors of vonBert param (Linf, K, t0), applied if params are estimated
70.42
0.1618
0.6215

##################################################################################################################
999 #end of data file flag
```