SEDAR

Southeast Data, Assessment, and Review

SEDAR 10

Stock Assessment Model

U.S. South Atlantic Gag Grouper

April, 2006

Erik H. Williams
Center for Coastal Fisheries and Habitat Research
Beaufort, North Carolina

Model 1 – Statistical Catch-at-age Model

1.1. General Modeling Approach

The essence of statistical catch-at-age models is to simulate a population that is projected forward in time like the population being assessed. Aspects of the fishing process (i.e., gear selectivity) are also simulated. Quantities to be estimated are systematically varied from starting values until the simulated populations characteristics match available data on the real population as closely as possible. Such data include total catch by fishery and year; observed age and length composition by gear and year; and observed indices of abundance.

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 model developed for this assessment is an elaboration of the CAGEAN and stock-synthesis models and very similar in structure to models used for assessment of Gulf of Mexico cobia (Williams 2001), U.S. South Atlantic red porgy (Anonymous 2002), U.S. South Atlantic black sea bass (Anonymous 2003), U.S. South Atlantic tilefish (Anonymous 2004), and U.S. South Atlantic snowy grouper (Anonymous 2004). Statistical catch-at-age models share many attributes with ADAPT-style tuned and untuned VPAs.

1.2. Methods

A general description of the assessment model follows.

Properties of age-structured model
The statistical catch-at-age model for this assessment was implemented in the AD Model Builder (ADMB) software (Otter Research 2001) on a microcomputer. A summary of the model equations are in Table 1. The formulation's major characteristics are summarized as follows:

Natural mortality rate - The natural mortality rate was assumed constant over time. A vector of age-specific *M* estimates based on Lorenzen (1996) was used as a starting estimate. The age-specific *M* vector was then re-scaled based on a fraction of survivors at the oldest age consistent with the findings of Hoenig (1983).

Stock dynamics - The standard Baranov catch equation was applied. This assumes exponential decay in population size due to fishing and natural mortality processes.

Growth/Maturity - Size and proportion female mature at age was assumed constant across years. Proportion female was allowed to vary over time based on data collected by the MARMAP survey. Gag grouper is a protogynous species and it was assumed that all males age-1 and older are fully mature.

Recruitment - A Beverton–Holt recruitment model was estimated internally. Estimated recruitments were loosely conditioned on that model.

Biological benchmarks - Biological benchmarks were calculated based on maximum sustainable yield (MSY) estimates from the Beverton–Holt recruitment model. These include the exploitation rate, fishing mortality rate, and total mature biomass at MSY (Emsy, Fmsy, and SSBmsy, respectively).

Fishing - Four fisheries were modeled individually: handline, diving, headboat, and shore/private boat/charter boat. Separate fishing mortality rates were estimated for each of these fisheries. Selectivity ata eg was allowed to vary by regulation period for the handline and headboat fisheries. The headboat selectivity was applied to the shore/private boat/charter boat fishery. Finally, the diving fishery only had enough data to estimate a single selectivity curve to all years.

Selectivity functions - Selectivity was fit parametrically, using a double-logistic model for the diving fishery and logistic models for the remaining fisheries, rather than estimating independent selectivity values for each age. That approach reduces the number of estimated parameters and imposes theoretical structure on the estimates.

Discards – Discards are available for the headboat, shore/private boat/charter boat, and handline fisheries. Diving discards are believed to be negligible and not included in this assessment.

Abundance indices -The model used three fishery dependent modeled indices of abundance. They include shore/private boat/charter boat CPUE (years 1981-2004) and handline CPUE (years 1992-2004) indices and headboat CPUE index (years 1973-2004).

Fitting criterion -The fitting criterion was a total likelihood approach in which fishery catch, observed age and length compositions, and the abundance index patterns were fit to the degree that they are compatible. Landings data and abundance index data were fit using a lognormal likelihood. Age and length composition data were fit using a multinomial likelihood. Relative statistical weightings of likelihood components for an initial model run were chosen at the assessment workshop after examining many candidate model runs. The criteria for choice were a balance of reasonable fit to all available data and a good degree of biological realism in estimated population trajectory.

Estimated Parameters
The model estimates many parameters.

Likelihood Component Weights
The selection of likelihood component weights for the initial run model involved an iterative process of model fitting, examination of the fit, and adjustment of the weights.

The performance of an individual model run was evaluated based on its fit to the observed datasets. These datasets include four time series of landings, three time series of discards, three abundance indices, and age and length compositions from both fishery and survey sources. The influence of each dataset on the overall model fit is determined by the specification of the error terms in each likelihood component. In the case of lognormal likelihoods, it is the annual coefficient of variation, and for the multinomial components, it is the annual sample sizes. These terms determine the influence of each year of data relative to other years of the same data source. However, the relative influence of different components can only be treated by re-weighting each likelihood. An objective determination of these weights is an unsolved problem in statistical modeling. In this case, the weights were determined by examination of overdispersion, model mis-specification (e.g. runs of residuals), and the general reliability (i.e. our understanding of information content) of the data source.

We reduced the number of weights to be examined by grouping likelihood components based on their type, scale, and method of collection. For example, the four fisheries landings data were grouped, so that a single weight was applied to all four components. Similarly the discard components were grouped, the index components were grouped, the age composition components were grouped, and the length composition components were grouped. The model also contains a likelihood component for the annual recruitment deviation parameters, which are constrained to follow a Beverton–Holt stock-recruit curve.

Table 1: **General definitions, input data, population model, and negative log-likelihood components of the forward-projecting statistical age-structured model used for gag grouper.**

General Definitions	**Symbol**	**Description/Definition**
Year index	y	y = {1942,..,2004}
Age index	a	a = {0,...,A}, where A = 20+
Length bin (mm)	l'	l' = {220,255,…,1220}, bin size = 30 mm
Fishery index	f	f = {1 handline, 2 diving, 3 headboat, 4 MRFSS}
CPUE index	u	u = {1 headboat CPUE, 2 MRFSS CPUE, 3 handline CPUE}
Input Data	**Symbol**	**Description/Definition**
Mean length-at-age	l_a	$l_a = L_\infty\left(1-\exp\left[-K\left(a-t_0\right)\right]\right)$ where parameters L_∞, K, and t_0 are fixed
Age-length conversion matrix	$\psi_{a,l'}$	$\psi_{a,l'} = \dfrac{\exp\left[-\left(\dfrac{l'-l_a}{2c^l l_a}\right)\right]}{\sqrt{2\pi}\left(c^l l_a\right)^2}$ where c^l is a fixed value for the coefficient of variation in length at age and the matrix is re-scaled to sum to 1 across ages
Population weight-at-age	w_a	Computed from size at age at the midpoint of the year $w_a = \gamma l_a^\beta$, where γ and β are fixed
Maturity-at-age	m_a	Logistic function of age, estimated from MARMAP sampled data
Observed CPUE indices	$U_{u,y}$	u=1, headboat (y = 1973,…,2004), based on numbers of fish captured per angler-day. u=2, MRFSS CPUE (y = 1981,…,2004), based on numbers of fish captured per angler-hour. u=3, handline CPUE(y = 1992,…,2004), based on weight of fish per .
Coefficient of variation for U 's	$c_{u,y}$	u = {1, 2, 3} (see above), annual values from GLM model or sampling error.
Observed age compositions	$p_{f,a,y}$	Computed as percent age composition at age (a) for each year (y) and fishery (f)
Age composition sample sizes	$n_{f,y}$	Number of age samples collected in each year (y) from each fishery (f)

Observed length compositions	$p'_{f,l,y}$	Computed as percent length composition at length (*l*) for each year (*y*) and fishery (*f*)
Length composition sample sizes	$n'_{f,y}$	Number of length samples collected in each year (*y*) from each fishery (*f*)
Observed fishery landings	$L_{f,y}$	Reported landings in weight for each year (*y*) from each fishery (*f*)
Coefficient of variation for L_f	$c_{L_f,y}$	Annual values fixed based on understanding of historical accuracy of estimates
Age-specific natural mortality	M_a	Fixed across years from Lorenzen (1996), re-scaled based on Hoenig (1983)

Population Model	**Symbol**	**Description/Definition**
Fishery selectivity	$s_{f,a}$	Constant for all years (*y*), except for headboat (*f* = 3) $$s_{f,a}=\begin{cases}\left[\dfrac{1}{1+\exp\left(-\eta_{1,f}\left[a-\alpha_{1,f}\right]\right)}\right] & \text{for } f=1\\ \left[\dfrac{1}{1+\exp\left(-\eta_{1,f}\left[a-\alpha_{1,f}\right]\right)}\right]\left[1-\dfrac{1}{1+\exp\left(-\eta_{2,f}\left[a-\left(\alpha_{1,f}+\alpha_{2,f}\right)\right]\right)}\right]\left[\dfrac{1}{\max\left(s_{f,a}\right)}\right] & \text{for f=\{2,3,4\}}\end{cases}$$ where $\eta_{1,f}$, $\eta_{2,f}$, $\alpha_{1,f}$ and $\alpha_{2,f}$ are estimated parameters, for headboat (*f* = 3), parameters $\alpha_{1,f}$ and $\alpha_{2,f}$ were multiplied by terms $\alpha_{3,f}$ and $\alpha_{4,f}$, respectively, for *y* = {1977,…,1991} to allow for a linear change over time and for MRFSS (*f* = 4), selectivity is fixed at the estimates from headboat in *y* = 2002.
Index selectivity	$s'_{u,a}$	Assumed constant for all years (*y*) $$s'_{u,a}=\left[\frac{1}{1+\exp\left(-\eta'_{1,u}\left[a-\alpha'_{1,u}\right]\right)}\right]\left[1-\frac{1}{1+\exp\left(-\eta'_{2,u}\left[a-\alpha'_{2,u}\right]\right)}\right]\left[\frac{1}{\max\left(s'_{u,a}\right)}\right] \quad \text{for } u=\{1,2\}$$ where $\eta'_{1,U}$, $\eta'_{2,U}$, $\alpha'_{1,U}$ and $\alpha'_{2,U}$ are estimated parameters, for *u* = 2, $s'_{u,a}=s_{1,a}$
Fishing mortality	$F_{f,a,y}$	$F_{f,a,y}=s_{f,a}F_{f,y}$ where $F_{f,y}$'s are fully selected estimated parameters
Total mortality	$Z_{a,y}$	$$Z_{a,y}=M_a+\sum_{f=1}^{4}F_{f,a,y}$$

Mature biomass per recruit at $F = 0$	ϕ_y	$\phi_y = \sum_{a=0}^{A} N_{a,y} m_a w_a \Big/ N_{0,y}$ where $N_{a+1,y} = N_{a,y} \exp(-Z_{a,y})$ and $N_{A,y} = N_{A-1,y} \exp(-Z_{A-1,y}) / [1-\exp(-Z_{A,y})]$
Population numbers Population mature biomass	$N_{a,y}$ S_y	$N_{0,1942} = R_0 + R_{1942}$ $N_{a+1,1942} = N_{a,1942} \exp(-Z_{a,1942})$ $N_{A,1942} = N_{A-1,1942} \exp(-Z_{A-1,1942}) / [1-\exp(-Z_{A,1942})]$ $N_{0,y} = \frac{0.8 R_0 h \varepsilon_y}{0.2\phi_y R_0 (1-h) + (h-0.2)\varepsilon_y} + R_y$ $N_{a+1,y+1} = N_{a,y} \exp(-Z_{a,y})$ $N_{A,y} = N_{A-1,y-1} \exp(-Z_{A-1,y-1}) + N_{A,y-1} \exp(-Z_{A,y-1})$ $S_y = \sum_{a=0}^{A} N_{a,y} m_a w_a$ where R_0 (virgin recruitment) and h (steepness) are parameters of the stock-recruit curve and R_y are annual recruitment deviation parameters.
Population biomass	B_y	$B_y = \sum_{a=0}^{A} N_{a,y} w_a$
Predicted catch-at-age	$\hat{C}_{f,a,y}$	$\hat{C}_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}} N_{a,y} [1-\exp(-Z_{a,y})]$
Predicted landings	$\hat{L}_{f,y}$	$\hat{L}_{f,y} = \sum_{a=0}^{A} \hat{C}_{f,a,y} w_a$
Predicted age composition	$\hat{p}_{\{f,u\},a,y}$	$\hat{p}_{\{f,u\},a,y} = \hat{C}_{\{f,u\},a,y} \Big/ \sum_{a=0}^{A} \hat{C}_{\{f,u\},a,y}$

Predicted CPUE indices	$\hat{U}_{u,y}$	$\hat{U}_{u,y} = \begin{cases} \sum_{a=0}^{A} N_{a,y} s'_{1,a} q_1 & \text{for } u = 1 \\ \sum_{a=0}^{A} N_{a,y} s'_{2,a} q_2 & \text{for } u = 2 \\ \sum_{a=0}^{A} N_{a,y} s_{3,a} q_3 & \text{for } u = 3 \end{cases}$ where q_1, q_2, and q_3 are catchability parameters

Negative Log-Likelihood	**Symbol**	**Description/Definition**
Multinomial age composition	Λ_1	$\Lambda_1 = -\lambda_1 n_{\{f,u\},y} \sum_{a=0}^{A} \left(p_{\{f,u\},a,y} + x\right) \log\left(\hat{p}_{\{f,u\},a,y} + x\right) - \left(p_{\{f,u\},a,y} + x\right) \log\left(p_{\{f,u\},a,y} + x\right)$ where λ_1 is a preset weighting factor and x is fixed at an arbitrary value of 0.001
Multinomial length composition	Λ_2	$\Lambda_2 = -\lambda_2 n'_{\{f,u\},y} \sum_{l'=225}^{1095} \left(p'_{\{f,u\},l',y} + x\right) \log\left(\hat{p}'_{\{f,u\},l',y} + x\right) - \left(p'_{\{f,u\},l',y} + x\right) \log\left(p'_{\{f,u\},l',y} + x\right)$ where λ_2 is a preset weighting factor and x is fixed at an arbitrary value of 0.001
Lognormal indices	Λ_3	$\Lambda_3 = \lambda_3 \sum_y \frac{\left[\log\left(U_{u,y} + x\right) - \log\left(\hat{U}_{u,y} + x\right)\right]^2}{2c^2_{u,y}}$ where λ_3 is a preset weighting factor and x is fixed at an arbitrary value of 0.001
Lognormal landings	Λ_4	$\Lambda_4 = \lambda_4 \sum_y \frac{\left[\log\left(L_{f,y} + x\right) - \log\left(\hat{L}_{f,y} + x\right)\right]^2}{2c^2_{L_f,y}}$ where λ_4 is a preset weighting factor and x is fixed at an arbitrary value of 0.001
Recruitment constraint	Λ_5	$\Lambda_5 = \lambda_5 \sum_y R_y^2$

Model 2 – PRODUCTION MODEL

1.3. Overview

An age-aggregated production model was also fit to available data. Production models are particularly useful when data are inadequate to classify individuals based on age or size. They are also a useful tool for exploration of management consequences because their relative simplicity makes it easier to understand the details of how manipulations are affecting results and performance. Their simplicity may also allow them to more powerfully fit observations that lack age or size structure, for example landings and abundance indices. However, the age or size structure of the population can give useful insight into its history and status. Consequently, when reliable data are available on the age, size, or both of individuals in a population, an age- or size-structured model can often be more informative. That is particularly true when data on relative abundance are uncertain or fragmented, as in this assessment.

Given the above, the workshop was hesitant to apply such a model this stock. Ultimately, the group decided that application of such a model should be examined in the course of the workshop, and that its results would have to pass critical examination before being accepted.

1.4. Methods (Production Model)

In this task, the Prager (1994) form of the Graham–Schaefer production model was used. This is a continuous time formulation, conditioned on catch, that does not assume equilibrium conditions. By conditioning on catch, the landings data are assumed more precise than the abundance indices. The model fits more than one abundance index by assuming they are correlated measures of stock abundance and that differences between indices can be considered sampling error. The Schaefer (1954; 1957) form of the production model, used here, assumes $B_{\mathrm{MSY}} = 0.5K$, where K is the carrying capacity of the stock (virgin stock size). The Schaefer form is often used as a default because of its theoretical simplicity and because it is considered a central case among possible shapes of production model. The ASPIC software of Prager (1995) was used.

Appendix I. ADMB code for the gag grouper model.

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR Update Assessment: Gag, May 2006
//##
//##  Erik Williams, NMFS, Beaufort Lab
//##  Erik.Williams@noaa.gov
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

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

!!cout << "Starting Gag Assessment Model" << endl;

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

//3 periods: until '91 no size regs, 1992-98 12inch TL, 1999-04 14inch TL
init_int endyr_period1;
init_int endyr_period2;

//Total number of ages
init_int nages;

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

//starting year for recruitment estimation (not being read in) and number assessment years
int styrR;
number nyrs;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
   styrR=styr-(nages-1);
   nyrs=endyr-styr+1.;
 END_CALCS

//Total number of length bins for each matrix
init_int nlenbins10;
//init_int nlenbins20;

// Vector of lengths for length bins (mm)(midpoint)
init_ivector lenbins10(1,nlenbins10);
//init_ivector lenbins20(1,nlenbins20);

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

//Total number of iterations for spr calcs
init_int n_iter_spr;
//Total number of iterations for msy calcs
init_int n_iter_msy;
//starting age for exploitation rate: ages are (value-1) to oldest
init_int set_E_age_st;
//bias correction (set to 1.0 for no bias correction or 0.0 to compute from rec variance)
init_number set_BiasCor;
// Von Bert parameters (from McGovern et al.)
init_number set_Linf;
init_number set_K;
init_number set_t0;
//CV of length at age
init_number set_len_cv;

//length(mm)-weight(gutted lbs) relationship: W=aL^b
init_number wgtpar_a;
init_number wgtpar_b;

//Sex ratio and maturity
```

```
init_matrix prop_m_obs(styr,endyr,1,nages);         //Proportion male by age
init_vector maturity_m_obs(1,nages);            //total maturity of males
init_vector maturity_f_obs(1,nages);            //total maturity of females

//###################Commercial Hook and Line fishery landings########################
//CPUE
init_int styr_HAL_cpue;
init_int endyr_HAL_cpue;
init_vector obs_HAL_cpue(styr_HAL_cpue,endyr_HAL_cpue);//Observed CPUE
init_vector HAL_cpue_cv(styr_HAL_cpue,endyr_HAL_cpue); //CV of cpue

// Landings (1000s gutted pounds)
init_int styr_commHAL_L;
init_int endyr_commHAL_L;
init_vector obs_commHAL_L(styr_commHAL_L,endyr_commHAL_L); //vector of observed landings by year
init_vector commHAL_L_cv(styr_commHAL_L,endyr_commHAL_L);    //vector of CV of landings by year

// Discards (1000s)
init_int styr_commHAL_D;
init_int endyr_commHAL_D;
init_vector obs_commHAL_released(styr_commHAL_D,endyr_commHAL_D); //vector of observed releases by year, multiplied by discard
mortality for fitting
init_vector commHAL_D_cv(styr_commHAL_D,endyr_commHAL_D);    //vector of CV of discards by year
// Length Compositions (30mm bins)
init_int styr_commHAL_lenc;
init_int endyr_commHAL_lenc;
init_vector nsamp_commHAL_lenc(styr_commHAL_lenc,endyr_commHAL_lenc);
init_matrix obs_commHAL_lenc(styr_commHAL_lenc,endyr_commHAL_lenc,1,nlenbins10);
// Age Compositions
init_int nyr_commHAL_agec;
init_ivector yrs_commHAL_agec(1,nyr_commHAL_agec);
init_vector nsamp_commHAL_agec(1,nyr_commHAL_agec);
init_matrix obs_commHAL_agec(1,nyr_commHAL_agec,1,nages);

//###############################Commercial Diving fishery fishery########################
// Landings (1000s gutted pounds)
init_int styr_commDV_L;
init_int endyr_commDV_L;
init_vector obs_commDV_L(styr_commDV_L,endyr_commDV_L);
init_vector commDV_L_cv(styr_commDV_L,endyr_commDV_L);    //vector of CV of landings by year
// Length Compositions (30mm bins)
init_int nyr_commDV_lenc;
init_ivector yrs_commDV_lenc(1,nyr_commDV_lenc);
init_vector nsamp_commDV_lenc(1,nyr_commDV_lenc);
init_matrix obs_commDV_lenc(1,nyr_commDV_lenc,1,nlenbins10);
// Age Compositions
init_int nyr_commDV_agec;
init_ivector yrs_commDV_agec(1,nyr_commDV_agec);
init_vector nsamp_commDV_agec(1,nyr_commDV_agec);
init_matrix obs_commDV_agec(1,nyr_commDV_agec,1,nages);

//################################Headboat landings########################################
//CPUE
init_int styr_HB_cpue;
init_int endyr_HB_cpue;
init_vector obs_HB_cpue(styr_HB_cpue,endyr_HB_cpue);//Observed CPUE
init_vector HB_cpue_cv(styr_HB_cpue,endyr_HB_cpue); //CV of cpue
// Landings (numbers, 1000s)
init_int styr_HB_L;
init_int endyr_HB_L;
init_vector obs_HB_L(styr_HB_L,endyr_HB_L);
init_vector HB_L_cv(styr_HB_L,endyr_HB_L);
// Discards (1000s)
init_int styr_HB_D;
init_int endyr_HB_D;
init_vector obs_HB_released(styr_HB_D,endyr_HB_D); //vector of observed releases by year, multiplied by discard mortality for fitting
init_vector HB_D_cv(styr_HB_D,endyr_HB_D);    //vector of CV of discards by year
// Length Compositions (10mm bins)
init_int styr_HB_lenc;
init_int endyr_HB_lenc;
init_vector nsamp_HB_lenc(styr_HB_lenc,endyr_HB_lenc);
init_matrix obs_HB_lenc(styr_HB_lenc,endyr_HB_lenc,1,nlenbins10);
// Age compositions
init_int nyr_HB_agec;
init_ivector yrs_HB_agec(1,nyr_HB_agec);
```

```
init_vector nsamp_HB_agec(1,nyr_HB_agec);
init_matrix obs_HB_agec(1,nyr_HB_agec,1,nages);

//############################MRFSS landings ##################################
//CPUE
init_int styr_MRFSS_cpue;
init_int endyr_MRFSS_cpue;
init_vector obs_MRFSS_cpue(styr_MRFSS_cpue,endyr_MRFSS_cpue);//Observed CPUE
init_vector MRFSS_cpue_cv(styr_MRFSS_cpue,endyr_MRFSS_cpue); //CV of cpue
// Landings (numbers, 1000s)
init_int styr_MRFSS_L;
init_int endyr_MRFSS_L;
init_vector obs_MRFSS_L(styr_MRFSS_L,endyr_MRFSS_L);
init_vector MRFSS_L_cv(styr_MRFSS_L,endyr_MRFSS_L);
// Discards (1000s)
init_int styr_MRFSS_D;
init_int endyr_MRFSS_D;
init_vector obs_MRFSS_released(styr_MRFSS_D,endyr_MRFSS_D); //vector of observed releases by year, multiplied by discard mortality for
fitting
init_vector MRFSS_D_cv(styr_MRFSS_D,endyr_MRFSS_D);    //vector of CV of discards by year

//##################Parameter values and initial guesses #################################
//--weights for likelihood components-------------------------------------------------------------------------------
init_number set_w_L;
init_number set_w_D;
init_number set_w_lc;
init_number set_w_ac;
init_number set_w_I_HAL;
init_number set_w_I_HB;
init_number set_w_I_MRFSS;
init_number set_w_R;
init_number set_w_R_init;
init_number set_w_R_end;
init_number set_w_F;
init_number set_w_B1dB0;        // weight on B1/B0
init_number set_w_fullF;        //penalty for any fullF>5
init_number set_w_cvlen_dev;        //penalty on cv deviations at age
init_number set_w_cvlen_diff;      //penalty on first difference of cv deviations at age

//Initial guess for commercial landings bias parameter
init_number set_L_commHAL_bias;
//Initial guess for rate of increase on q
init_number set_q_rate;
//Initial guesses or fixed values
init_number set_steep;
//init_number set_M;
init_vector set_M(1,nages);

//--index catchability------------------------------------------------------------------------------------------------------------
init_number set_logq_HAL;     //catchability coefficient (log) for commercial logbook CPUE index
init_number set_logq_HB;      //catchability coefficient (log) for the headboat index
init_number set_logq_MRFSS;   //catchability coefficient (log) for MRFSS CPUE index

//--F's--------------------------------
init_number set_log_avg_F_commHAL;
init_number set_log_avg_F_commDV;
init_number set_log_avg_F_HB;
init_number set_log_avg_F_MRFSS;

//--discard F's-----------------------
init_number set_log_avg_F_commHAL_D;
init_number set_log_avg_F_HB_D;
init_number set_log_avg_F_MRFSS_D;

//Set some more initial guesses of estimated parameters
init_number set_log_R0;
init_number set_S1dS0;
init_number set_R1_mult;
init_number set_B1dB0;

//Initial guesses of estimated selectivity parameters
init_number set_selpar_L50_commHAL1;
init_number set_selpar_slope_commHAL1;
init_number set_selpar_L50_commHAL2;
init_number set_selpar_slope_commHAL2;
init_number set_selpar_L50_commHAL3;
init_number set_selpar_slope_commHAL3;

init_number set_selpar_L50_commDV1;
init_number set_selpar_L502_commDV1;
init_number set_selpar_slope_commDV1;
init_number set_selpar_slope2_commDV1;
//init_number set_selpar_L50_commDV2;
//init_number set_selpar_L502_commDV2;
//init_number set_selpar_slope_commDV2;
//init_number set_selpar_slope2_commDV2;

init_number set_selpar_L50_HB1;
init_number set_selpar_slope_HB1;
init_number set_selpar_L50_HB2;
init_number set_selpar_slope_HB2;
init_number set_selpar_L50_HB3;
init_number set_selpar_slope_HB3;

// #######Indices for year(iyear), age(iage),length(ilen) ###############
int iyear;
int iage;
int ilen10;
int E_age_st;   //starting age for exploitation rate: (value-1) to oldest

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 IN CORRECTLY ****" << endl;
       cout << "" <<endl;
     }
   }
 END_CALCS

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

  init_bounded_number Linf(600,1400,2);
  init_bounded_number K(0.05,0.6,2);
  init_bounded_number t0(-2.0,0.0,2);
  vector wgt(1,nages);
  vector meanlen(1,nages);                           //mean length at age
  number sqrt2pi;
  matrix lenprob10(1,nages,1,nlenbins10);            //distn of size at age (age-length key, 10 mm bins)
  //init_bounded_vector len_cv(1,nages,0.01,0.5,3); //cv of length at age
  init_bounded_number log_len_cv(-4.6,-0.7,2) //cv expressed in log-space, bounds correspond to 0.01, 0.5
  init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
  vector len_cv(1,nages);

//----Age and length compositions
  matrix pred_commHAL_lenc(styr_commHAL_lenc,endyr_commHAL_lenc,1,nlenbins10);
  matrix pred_commDV_lenc(1,nyr_commDV_lenc,1,nlenbins10);
  matrix pred_HB_lenc(styr_HB_lenc,endyr_HB_lenc,1,nlenbins10);
  matrix pred_commHAL_agec(1,nyr_commHAL_agec,1,nages);
  matrix pred_commDV_agec(1,nyr_commDV_agec,1,nages);
  matrix pred_HB_agec(1,nyr_HB_agec,1,nages);

  //nsamp_X_allyr vectors used only for R output of comps with nonconsecutive yrs
  vector nsamp_commDV_lenc_allyr(styr,endyr);
  vector nsamp_commHAL_agec_allyr(styr,endyr);
  vector nsamp_commDV_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);

//-----Population--------------------------------------------------------------------------------
  matrix N(styrR,endyr+1,1,nages);          //Population numbers by year and age
  matrix B(styrR,endyr+1,1,nages);          //Population biomass by year and age
  vector totB(styrR,endyr+1);               //Total biomass by year
  number R1;                                //Recruits in styrR
  //init_bounded_number log_R1(5,20,1);       //log(Recruits) in styrR
  sdreport_vector SSB(styrR,endyr+1);       //Spawning biomass by year
  sdreport_vector rec(styrR,endyr+1);       //Recruits by year
  matrix prop_m(styrR,endyr,1,nages);       //Proportion male by age
  matrix prop_f(styrR,endyr,1,nages);        //Proportion female by age
  matrix maturity_f(styrR,endyr,1,nages);
  matrix maturity_m(styrR,endyr,1,nages);  //time-invariant, but left with flexibility to change that
  matrix reprod(styrR,endyr,1,nages);

//---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(5,20,1);       //log(virgin Recruitment)
  sdreport_number R0;
  init_bounded_number steep(0.25,0.95,1);     //steepness
  //number steep;  //uncomment to fix steepness, comment line directly above
  init_bounded_dev_vector log_dev_N_rec(styrR+1,endyr,-3,3,2); //log recruitment deviations
  number var_rec_dev;                       //variance of log recruitment deviations.
                                    //Estimated from yrs with unconstrainted S-R(1972-2001)
  number BiasCor;                          //Bias correction in equilibrium recruits
  sdreport_number steep_sd;                 //steepness for stdev report
  number S0;                              //equal to spr_F0*R0 = virgin SSB
  number B0;                              //equal to bpr_F0*R0 = virgin B
  number S1;                              //initial SSB
  number S1dS0;                            //S1967/S0
  number B1dB0;                            //B1dB0 computed and used in constraint
  init_bounded_number R1_mult(0.5,1.5,1);     //R1967=R1_mult*R0
  sdreport_number S1S0;                     //SSB(styr) / virgin SSB
  sdreport_number popstatus;                //SSB(endyr) / virgin SSB

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

  //Commercial hook and line
  matrix sel_commHAL(styrR,endyr,1,nages);
  init_bounded_number selpar_slope_commHAL1(0.5,9.0,1); //period 1
  init_bounded_number selpar_L50_commHAL1(1.0,10.0,1);
  init_bounded_number selpar_slope_commHAL2(0.5,9.0,1); //period 2
  init_bounded_number selpar_L50_commHAL2(1.0,10,1);
  init_bounded_number selpar_slope_commHAL3(0.5,9.0,1); //period 3
  init_bounded_number selpar_L50_commHAL3(1.0,10,1);
  init_bounded_dev_vector selpar_L50_commHAL_dev(styr_commHAL_lenc,endyr_period1,-5,5,3);
  //Commercial diving
  matrix sel_commDV(styrR,endyr,1,nages);            //period 1
  init_bounded_number selpar_slope_commDV1(0.5,9.0,1);
  init_bounded_number selpar_L50_commDV1(1.0,10,1);
  init_bounded_number selpar_slope2_commDV1(0.1,9.0,1);
  init_bounded_number selpar_L502_commDV1(1.0,20.0,1);//period 2
  //Headboat: logistic, parameters allowed to vary with period defined by size restrictions
  matrix sel_HB(styrR,endyr,1,nages);
  init_bounded_number selpar_slope_HB1(0.5,9.0,1);     //period 1
  init_bounded_number selpar_L50_HB1(0.1,10.0,1);
  init_bounded_number selpar_slope_HB2(0.5,9.0,1);     //period 2
  init_bounded_number selpar_L50_HB2(0.1,10.0,1);
  init_bounded_number selpar_slope_HB3(0.5,9.0,1);     //period 3
  init_bounded_number selpar_L50_HB3(0.1,10.0,1);
  init_bounded_dev_vector selpar_L50_HB_dev(styr_HB_lenc,endyr_period1,-5,5,3);
  //MRFSS: same as HB selectivity (AW)
  matrix sel_MRFSS(styrR,endyr,1,nages);

  //effort-weighted, recent selectivities
  vector sel_wgted_L(1,nages);  //toward landings
  vector sel_wgted_D(1,nages);  //toward discards
  vector sel_wgted_tot(1,nages);//toward Z, landings plus deads discards
  number max_sel_wgted_tot;

//-------CPUE Predictions-------------------------------
  vector pred_HAL_cpue(styr_HAL_cpue,endyr_HAL_cpue);          //predicted HAL U (pounds/hook-hour)
  matrix N_HAL(styr_HAL_cpue,endyr_HAL_cpue,1,nages);          //used to compute HAL index
  vector pred_HB_cpue(styr_HB_cpue,endyr_HB_cpue);             //predicted HB U (number/angler-day)
  matrix N_HB(styr_HB_cpue,endyr_HB_cpue,1,nages);             //used to compute HB index
  vector pred_MRFSS_cpue(styr_MRFSS_cpue,endyr_MRFSS_cpue);    //predicted MRFSS U (number/1000 hook-hours)
  matrix N_MRFSS(styr_MRFSS_cpue,endyr_MRFSS_cpue,1,nages);    //used to compute MRFSS index

//---Catchability (CPUE q's)--------------------------------------------------------
  init_bounded_number log_q_HAL(-20,-5,1);
  init_bounded_number log_q_HB(-20,-5,1);
  init_bounded_number log_q_MRFSS(-20,-5,1);
  init_bounded_number q_rate(-0.1,0.1,-3);

//---Landings Bias-------------------------------------------------------------------
  init_bounded_number L_commHAL_bias(0.1,5.0,3);

//---Catch (numbers), Landings (mt)--------------------------------------------------
  matrix C_commHAL(styrR,endyr,1,nages);                //catch (numbers) at age
  matrix L_commHAL(styrR,endyr,1,nages);                //landings (mt) at age
  vector pred_commHAL_L(styr_commHAL_L,endyr_commHAL_L); //yearly landings summed over ages

  matrix C_commDV(styrR,endyr,1,nages);                 //catch (numbers) at age
  matrix L_commDV(styrR,endyr,1,nages);                 //landings (mt) at age
  vector pred_commDV_L(styr_commDV_L,endyr_commDV_L); //yearly landings summed over ages

  matrix C_HB(styrR,endyr,1,nages);                     //catch (numbers) at age
  matrix L_HB(styrR,endyr,1,nages);                     //landings (mt) at age
  vector pred_HB_L(styr_HB_L,endyr_HB_L);               //yearly landings summed over ages

  matrix C_MRFSS(styrR,endyr,1,nages);                  //catch (numbers) at age
  matrix L_MRFSS(styrR,endyr,1,nages);                  //landings (mt) at age
  vector pred_MRFSS_L(styr_MRFSS_L,endyr_MRFSS_L);      //yearly landings summed over ages

  matrix C_total(styrR,endyr,1,nages);
  matrix L_total(styrR,endyr,1,nages);
  vector L_total_yr(styrR,endyr);                       //total landings by yr summed over ages

//---Discards (number dead fish) --------------------------------------------------
  matrix C_commHAL_D(styr_commHAL_D,endyr_commHAL_D,1,nages);//discards (numbers) at age
  vector pred_commHAL_D(styr_commHAL_D,endyr_commHAL_D);     //yearly discards summed over ages
  vector obs_commHAL_D(styr_commHAL_D,endyr_commHAL_D);      //observed releases multiplied by discard mortality

  matrix C_HB_D(styr_HB_D,endyr_HB_D,1,nages);           //discards (numbers) at age
  vector pred_HB_D(styr_HB_D,endyr_HB_D);                //yearly discards summed over ages
  vector obs_HB_D(styr_HB_D,endyr_HB_D);                 //observed releases multiplied by discard mortality

  matrix C_MRFSS_D(styr_HB_D,endyr_MRFSS_D,1,nages);     //discards (numbers) at age
  vector pred_MRFSS_D(styr_HB_D,endyr_MRFSS_D);          //yearly discards summed over ages
  vector obs_MRFSS_D(styr_MRFSS_D,endyr_MRFSS_D);        //observed releases multiplied by discard mortality

//---MSY calcs-----------------------------------------------------------------------------

  number F_commHAL_prop;  //proportion of F_full attributable to hal, last three yrs
  number F_commDV_prop;  //proportion of F_full attributable to diving, last three yrs
  number F_HB_prop;       //proportion of F_full attributable to headboat, last three yrs
  number F_MRFSS_prop;    //proportion of F_full attributable to MRFSS, last three yrs
  number F_commHAL_D_prop;//proportion of F_full attributable to hal discards, last three yrs
  number F_HB_D_prop;     //proportion of F_full attributable to headboat discards, last three yrs
  number F_MRFSS_D_prop;  //proportion of F_full attributable to MRFSS discards, last three yrs
  number F_temp_sum;      //sum of geom mean full Fs in last yrs, used to compute F_fishery_prop

  number SSB_msy_out;           //SSB at msy
  number F_msy_out;             //F at msy
  number msy_out;               //max sustainable yield
  number B_msy_out;             //total biomass at MSY
  number E_msy_out;             //exploitation rate (age 1+) at MSY
  number R_msy_out;             //equilibrium recruitment at F=Fmsy
  number D_msy_out;             //equilibrium dead discards at F=Fmsy
  number spr_msy_out;           //spr at F=Fmsy

  vector N_age_msy(1,nages);         //numbers at age for MSY calculations
  vector C_age_msy(1,nages);         //catch at age for MSY calculations
  vector Z_age_msy(1,nages);         //total mortality at age for MSY calculations
  vector D_age_msy(1,nages);         //discard mortality (dead discards) at age for MSY calculations
  vector F_L_age_msy(1,nages);       //fishing mortality (landings, not discards) at age for MSY calculations
  vector F_D_age_msy(1,nages);
  vector F_msy(1,n_iter_msy);        //values of full F to be used in per-recruit and 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(1,n_iter_msy);     //equilibrium landings(mt) 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 E_eq(1,n_iter_msy);     //equilibrium exploitation rates corresponding to F values in F_msy
  vector D_eq(1,n_iter_msy);     //equilibrium discards (1000s) corresponding to F values in F_msy

  vector FdF_msy(styrR,endyr);
  vector EdE_msy(styrR,endyr);
  vector SdSSB_msy(styrR,endyr+1);
  number SdSSB_msy_end;
  number FdF_msy_end;
  number EdE_msy_end;

//--------Mortality------------------------------------------------------------------
  vector M(1,nages);
  matrix F(styrR,endyr,1,nages);
  vector fullF(styrR,endyr);                    //Fishing mortality rate by year
  vector E(styrR,endyr);                        //Exploitation rate by year
  sdreport_vector fullF_sd(styrR,endyr);
  sdreport_vector E_sd(styrR,endyr);
  matrix Z(styrR,endyr,1,nages);

  init_bounded_number log_avg_F_commHAL(-10,0,1);
  init_bounded_dev_vector log_F_dev_commHAL(styr_commHAL_L,endyr_commHAL_L,-10,5,1);
  matrix F_commHAL(styrR,endyr,1,nages);
  vector F_commHAL_out(styrR,endyr_commHAL_L); //used for intermediate calculations in fcn get_mortality
  number log_F_init_commHAL;

  init_bounded_number log_avg_F_commDV(-10,0,1);
  init_bounded_dev_vector log_F_dev_commDV(styr_commDV_L,endyr_commDV_L,-10,5,2);
  matrix F_commDV(styrR,endyr,1,nages);
  vector F_commDV_out(styrR,endyr_commDV_L); //used for intermediate calculations in fcn get_mortality
  number log_F_init_commDV;

  init_bounded_number log_avg_F_HB(-10,0,1);
  init_bounded_dev_vector log_F_dev_HB(styr_HB_L,endyr_HB_L,-10,5,2);
  matrix F_HB(styrR,endyr,1,nages);
  vector F_HB_out(styrR,endyr_HB_L);        //used for intermediate calculations in fcn get_mortality
  number log_F_init_HB;

  init_bounded_number log_avg_F_MRFSS(-10,0,1);
  init_bounded_dev_vector log_F_dev_MRFSS(styr_MRFSS_L,endyr_MRFSS_L,-10,5,2);
  matrix F_MRFSS(styrR,endyr,1,nages);
  vector F_MRFSS_out(styrR,endyr_MRFSS_L); //used for intermediate calculations in fcn get_mortality
  number log_F_init_MRFSS;

//--Discard mortality stuff------------------------------------------------------------------------------
  init_bounded_number log_avg_F_commHAL_D(-10,0,1);
  init_bounded_dev_vector log_F_dev_commHAL_D(styr_commHAL_D,endyr_commHAL_D,-10,5,2);
  matrix F_commHAL_D(styrR,endyr,1,nages);
  vector F_commHAL_D_out(styr_commHAL_D,endyr_commHAL_D); //used for intermediate calculations in fcn get_mortality
  matrix sel_commHAL_D(styrR,endyr,1,nages);

  init_bounded_number log_avg_F_HB_D(-10,0,1);
  init_bounded_dev_vector log_F_dev_HB_D(styr_HB_D,endyr_HB_D,-10,5,2);
  matrix F_HB_D(styrR,endyr,1,nages);
  vector F_HB_D_out(styr_HB_D,endyr_HB_D); //used for intermediate calculations in fcn get_mortality
  matrix sel_HB_D(styrR,endyr,1,nages);

  init_bounded_number log_avg_F_MRFSS_D(-10,0,1);
  init_bounded_dev_vector log_F_dev_MRFSS_D(styr_MRFSS_D,endyr_MRFSS_D,-10,5,2);
  matrix F_MRFSS_D(styrR,endyr,1,nages);
  vector F_MRFSS_D_out(styr_MRFSS_D,endyr_MRFSS_D); //used for intermediate calculations in fcn get_mortality
  matrix sel_MRFSS_D(styrR,endyr,1,nages);

  number Dmort_commHAL;
  number Dmort_HB;
  number Dmort_MRFSS;

////---Per-recruit stuff----------------------------------------------------------------------------------
  vector N_age_spr(1,nages);         //numbers at age for SPR calculations
  vector C_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(styrR,endyr);    //vector of static SPR values by year
  vector F_L_age_spr(1,nages);       //fishing mortality (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 and equilibrium calculations
  vector spr_spr(1,n_iter_spr);      //reporductive capacity-per-recruit values corresponding to F values in F_spr
  vector L_spr(1,n_iter_spr);        //landings(mt)-per-recruit values corresponding to F values in F_spr
  vector E_spr(1,n_iter_spr);        //exploitation rate values corresponding to F values in F_spr

  vector N_spr_F0(1,nages);          //Used to compute spr at F=0
  vector spr_F0(styrR,endyr);        //Spawning biomass per recruit at F=0
  vector bpr_F0(styrR,endyr);        //Biomass per recruit at F=0

//-------Objective function components-----------------------------------------------------------------------------
  number w_L;
  number w_D;
  number w_lc;
  number w_ac;
  number w_I_HAL;
  number w_I_HB;
  number w_I_MRFSS;
  number w_R;
  number w_R_init;
  number w_R_end;
  number w_F;
  number w_B1dB0;
  number w_fullF;
  number w_cvlen_dev;
  number w_cvlen_diff;

  number f_HAL_cpue;
  number f_HB_cpue;
  number f_MRFSS_cpue;

  number f_commHAL_L;
  number f_commDV_L;
  number f_HB_L;
  number f_MRFSS_L;

  number f_commHAL_D;
  number f_HB_D;
  number f_MRFSS_D;

  number f_commHAL_lenc;
  number f_commDV_lenc;
  number f_HB_lenc;

  number f_commHAL_agec;
  number f_commDV_agec;
  number f_HB_agec;

  number f_N_dev;           //weight on recruitment deviations to fit S-R curve
  number f_N_dev_early;     //extra weight against deviations before styr
  number f_N_dev_last3;     //extra constraint on last 3 years of recruitment variability
  number f_Fend_constraint; //penalty for F deviation in last 5 years
  number f_B1dB0_constraint;//penalty to fix B(1967)/K
  number f_fullF_constraint;//penalty for fullF>5
  number f_cvlen_dev_constraint; //deviation penalty on cv's of length at age
  number f_cvlen_diff_constraint; //first diff penalty on cv's of length at age

  objective_function_value fval;
  number fval_unwgt;


//--Dummy arrays for output convenience  --------------------------
  vector xdum(styrR,endyr);
  vector xdum2(styrR,endyr+1);
//--Other dummy variables ----
  number sel_diff_dum;
  number zero_dum;
  number dzero_dum;

   //init_number x_dum; //used only during model development. can be removed.

//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
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 500, 2000, 10000;
 convergence_criteria 1e-1, 1e-2, 1e-4;

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

// Set values of fixed parameters or set initial guess of estimated parameters
  Dmort_commHAL=set_Dmort_commHAL;
  Dmort_HB=set_Dmort_HB;
  Dmort_MRFSS=set_Dmort_MRFSS;

  obs_commHAL_D=Dmort_commHAL*obs_commHAL_released;
  obs_HB_D=Dmort_HB*obs_HB_released;
  obs_MRFSS_D=Dmort_MRFSS*obs_MRFSS_released;

  E_age_st=set_E_age_st;   //E computed over (E_age_st-1)+   [minus 1 bc model starts with age 0]

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

  M=set_M;
  steep=set_steep;
  log_dev_N_rec=0.0;

  log_q_HAL=set_logq_HAL;
  log_q_HB=set_logq_HB;
  log_q_MRFSS=set_logq_MRFSS;
  q_rate=set_q_rate;

  L_commHAL_bias=set_L_commHAL_bias;

  w_L=set_w_L;
  w_D=set_w_D;
  w_lc=set_w_lc;
  w_ac=set_w_ac;
  w_I_HAL=set_w_I_HAL;
  w_I_HB=set_w_I_HB;
  w_I_MRFSS=set_w_I_MRFSS;
  w_R=set_w_R;
  w_R_init=set_w_R_init;
  w_R_end=set_w_R_end;
  w_F=set_w_F;
  w_B1dB0=set_w_B1dB0;
  w_fullF=set_w_fullF;
  w_cvlen_dev=set_w_cvlen_dev;
  w_cvlen_diff=set_w_cvlen_diff;

  log_avg_F_commHAL=set_log_avg_F_commHAL;
  log_avg_F_commDV=set_log_avg_F_commDV;
  log_avg_F_HB=set_log_avg_F_HB;
  log_avg_F_MRFSS=set_log_avg_F_MRFSS;

  log_avg_F_commHAL_D=set_log_avg_F_commHAL_D;
  log_avg_F_HB_D=set_log_avg_F_HB_D;
  log_avg_F_MRFSS_D=set_log_avg_F_MRFSS_D;

  log_len_cv=log(set_len_cv);
  log_R0=set_log_R0;
  S1dS0=set_S1dS0;
  R1_mult=set_R1_mult;
  B1dB0=set_B1dB0;

  selpar_L50_commHAL1=set_selpar_L50_commHAL1;
  selpar_slope_commHAL1=set_selpar_slope_commHAL1;
  selpar_L50_commHAL2=set_selpar_L50_commHAL2;
  selpar_slope_commHAL2=set_selpar_slope_commHAL2;
  selpar_L50_commHAL3=set_selpar_L50_commHAL3;
  selpar_slope_commHAL3=set_selpar_slope_commHAL3;

  selpar_L50_commDV1=set_selpar_L50_commDV1;
  selpar_L502_commDV1=set_selpar_L502_commDV1;
  selpar_slope_commDV1=set_selpar_slope_commDV1;
  selpar_slope2_commDV1=set_selpar_slope2_commDV1;
  //selpar_L50_commDV2=set_selpar_L50_commDV2;
  //selpar_L502_commDV2=set_selpar_L502_commDV2;
  //selpar_slope_commDV2=set_selpar_slope_commDV2;
  //selpar_slope2_commDV2=set_selpar_slope2_commDV2;

  selpar_L50_HB1=set_selpar_L50_HB1;
  selpar_slope_HB1=set_selpar_slope_HB1;
  selpar_L50_HB2=set_selpar_L50_HB2;
  selpar_slope_HB2=set_selpar_slope_HB2;
  selpar_L50_HB3=set_selpar_L50_HB3;
  selpar_slope_HB3=set_selpar_slope_HB3;

 sqrt2pi=sqrt(2.*3.14159265);
 //df=0.001; //difference for msy derivative approximations
 zero_dum=0.0;

 //additive constant to prevent division by zero
 dzero_dum=0.001;

 SSB_msy_out=0.0;

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

//Fill in sample sizes of comps sampled in nonconsec yrs.
//Used only for output in R object
      nsamp_commDV_lenc_allyr=missing; //"missing" defined in admb2r.cpp
      nsamp_commHAL_agec_allyr=missing;
      nsamp_commDV_agec_allyr=missing;
      nsamp_HB_agec_allyr=missing;
      for (iyear=1; iyear<=nyr_commDV_lenc; iyear++)
        {
          nsamp_commDV_lenc_allyr(yrs_commDV_lenc(iyear))=nsamp_commDV_lenc(iyear);
        }
      for (iyear=1; iyear<=nyr_commHAL_agec; iyear++)
        {
          nsamp_commHAL_agec_allyr(yrs_commHAL_agec(iyear))=nsamp_commHAL_agec(iyear);
        }
      for (iyear=1; iyear<=nyr_commDV_agec; iyear++)
        {
          nsamp_commDV_agec_allyr(yrs_commDV_agec(iyear))=nsamp_commDV_agec(iyear);
        }
      for (iyear=1; iyear<=nyr_HB_agec; iyear++)
        {
          nsamp_HB_agec_allyr(yrs_HB_agec(iyear))=nsamp_HB_agec(iyear);
        }



//fill in F's and Catch matrices with zero's
  F_commHAL.initialize();
  C_commHAL.initialize();
  F_commDV.initialize();
  C_commDV.initialize();
  F_HB.initialize();
  C_HB.initialize();
  F_MRFSS.initialize();
  C_MRFSS.initialize();

  F_commHAL_D.initialize();
  F_HB_D.initialize();
  F_MRFSS_D.initialize();

  sel_commHAL_D.initialize();
  sel_HB_D.initialize();
  sel_MRFSS_D.initialize();

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


//>--><>--><>--><>--><>
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
PROCEDURE_SECTION
 R0=mfexp(log_R0);

 //cout<<"start"<<endl;
 get_length_and_weight_at_age();
 get_reprod();
 //cout << "got length and weight transitions" <<endl;
 get_length_at_age_dist();
 //cout<< "got predicted length at age distribution"<<endl;
 get_spr_F0();
 //cout << "got F0 spr" << endl;
 get_selectivity();
 //cout << "got selectivity" << endl;
 get_mortality();
 //cout << "got mortalities" << endl;
 get_numbers_at_age();
 //cout << "got numbers at age" << endl;
 get_catch();
 //cout << "got catch at age" << endl;
 get_landings();
 //cout << "got landings" << endl;
 get_discards();
 //cout << "got discards" << 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;

 if(last_phase())
 {
   cout << "L_commHAL_bias = " << L_commHAL_bias << endl;
 }

FUNCTION get_length_and_weight_at_age
  //compute mean length (mm) and weight (gutted pounds) at age
  for (iage=1;iage<=nages;iage++)
  {
    meanlen(iage)=Linf*(1.0-mfexp(-K*((agebins(iage)+0.5)-t0)));
    wgt(iage)=0.001*wgtpar_a*pow(meanlen(iage),wgtpar_b); //.001 converts from gutted pounds to 1000 gutted pounds
  }

FUNCTION get_reprod
  for (iyear=styrR;iyear<=endyr;iyear++)
  {
   //product of stuff going into reproductive capacity calcs
   reprod(iyear)=elem_prod((elem_prod(prop_f(iyear),maturity_f(iyear))+elem_prod(prop_m(iyear),maturity_m(iyear))),wgt);
  }

FUNCTION get_length_at_age_dist
  //compute matrix of length at age, based on the normal distribution
  for (iage=1;iage<=nages;iage++)
  {
    len_cv(iage)=mfexp(log_len_cv+log_len_cv_dev(iage));
    for (ilen10=1;ilen10<=nlenbins10;ilen10++)
    {
      lenprob10(iage,ilen10)=(mfexp(-(square(lenbins10(ilen10)-meanlen(iage))/
      (2.*square(len_cv(iage)*meanlen(iage)))))/(sqrt2pi*len_cv(iage)*meanlen(iage)));
    }
    lenprob10(iage)/=sum(lenprob10(iage)); //standardize to account for truncated normal (i.e., no sizes<0)

  }

FUNCTION get_spr_F0
  N_spr_F0(1)=1.0;
  for (iage=2; iage<=nages; iage++)
  {
    N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*M(iage-1));
  }
  N_spr_F0(nages)=N_spr_F0(nages-1)*mfexp(-1.0*M(nages-1))/(1.0-mfexp(-1.0*M(nages-1))); //plus group

  for(iyear=styrR; iyear<=endyr; iyear++)
  {
    //spr_F0(iyear)=sum(elem_prod( elem_prod(elem_prod(N_spr_F0,prop_f),maturity_f(iyear))+
    //elem_prod(elem_prod(N_spr_F0,prop_m),maturity_m(iyear)) ,wgt));
    spr_F0(iyear)=sum(elem_prod(N_spr_F0,reprod(iyear)));
    bpr_F0(iyear)=sum(elem_prod(N_spr_F0,wgt));
  }


FUNCTION get_selectivity

//---time-varying selectivities

  for (iyear=styrR; iyear<=endyr_period1; iyear++)
  {
     for (iage=1; iage<=nages; iage++)
     {
       sel_HB(iyear,iage)=1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))-selpar_L50_HB1))); //logistic

       sel_commHAL(iyear,iage)=1./(1.+mfexp(-1.*selpar_slope_commHAL1*(double(agebins(iage))-selpar_L50_commHAL1))); //logistic

       sel_commDV(iyear,iage)=(1./(1.+mfexp(-1.*selpar_slope_commDV1*(double(agebins(iage))-
                     selpar_L50_commDV1))))*(1-(1./(1.+mfexp(-1.*selpar_slope2_commDV1*
                     (double(agebins(iage))-(selpar_L50_commDV1+selpar_L502_commDV1)))))); //double logistic
     }
     if (iyear>=styr_HB_lenc)
     {
       for (iage=1; iage<=nages; iage++)
       {
         sel_HB(iyear,iage)=1./(1.+mfexp(-1.*selpar_slope_HB1*(double(agebins(iage))
              -(selpar_L50_HB1+selpar_L50_HB_dev(iyear))))); //logistic
       }
     }
     if (iyear>=styr_commHAL_lenc)
     {
       for (iage=1; iage<=nages; iage++)
       {
         sel_commHAL(iyear,iage)=1./(1.+mfexp(-1.*selpar_slope_commHAL1*(double(agebins(iage))
              -(selpar_L50_commHAL1+selpar_L50_commHAL_dev(iyear))))); //logistic
       }
     }
     sel_commDV(iyear)=sel_commDV(iyear)/max(sel_commDV(iyear)); //re-normalize double logistic
  }


  for (iyear=endyr_period1+1; iyear<=endyr_period2; iyear++)
  {
     for (iage=1; iage<=nages; iage++)
     {
       sel_HB(iyear,iage)=1./
        (1.+mfexp(-1.*selpar_slope_HB2*(double(agebins(iage))-selpar_L50_HB2))); //logistic

       sel_commHAL(iyear,iage)=1./
        (1.+mfexp(-1.*selpar_slope_commHAL2*(double(agebins(iage))-selpar_L50_commHAL2))); //logistic

       sel_commDV(iyear,iage)=(1./(1.+mfexp(-1.*selpar_slope_commDV1*(double(agebins(iage))-
                     selpar_L50_commDV1))))*(1.-(1./(1.+mfexp(-1.*selpar_slope2_commDV1*
```

```
                    (double(agebins(iage))-(selpar_L50_commDV1+selpar_L502_commDV1))))));
   }
   sel_commDV(iyear)=sel_commDV(iyear)/max(sel_commDV(iyear));

 }

 for (iyear=endyr_period2+1; iyear<=endyr; iyear++)
 {
   for (iage=1; iage<=nages; iage++)
   {
     sel_HB(iyear,iage)=1./
      (1.+mfexp(-1.*selpar_slope_HB3*(double(agebins(iage))-selpar_L50_HB3))); //logistic
     sel_commHAL(iyear,iage)=1./
      (1.+mfexp(-1.*selpar_slope_commHAL3*(double(agebins(iage))-selpar_L50_commHAL3))); //logistic
   }
   sel_commDV(iyear)=sel_commDV(endyr_period2); //period3 sel same as period2 sel
  }

  //Discard selectivities

  for (iyear=styr_HB_D;iyear<=endyr_HB_D;iyear++)
  {
    if(iyear<=endyr_period2)
    {
      for (iage=1; iage<=nages; iage++)
      {
        sel_HB_D(iyear,iage)=(max(column(sel_HB, iage))-sel_HB(endyr_period2,iage));
      }
      sel_HB_D(iyear)=sel_HB_D(iyear)/(max(sel_HB_D(iyear))+dzero_dum); //prevent division by zero
    }
    else
    {
      for (iage=1; iage<=nages; iage++)
      {
        sel_HB_D(iyear,iage)=(max(column(sel_HB, iage))-sel_HB(endyr,iage));
      }
      sel_HB_D(iyear)=sel_HB_D(iyear)/(max(sel_HB_D(iyear))+dzero_dum); //prevent division by zero
    }
  }
  sel_MRFSS=sel_HB;
  sel_MRFSS_D=sel_HB_D;

  //for (iyear=styr_HB_D;iyear<=endyr_HB_D;iyear++)
  //{
  //   if(iyear<=endyr_period2)
  //   {
  //     for (iage=1; iage<=(nages-2); iage++)
  //     {
  //       sel_HB_D(iyear,iage)=(sel_HB(endyr_period2,iage+2)-sel_HB(endyr_period2,iage));
  //     }
  //     sel_HB_D(iyear,(nages-1))=0.0;
  //     sel_HB_D(iyear,nages)=0.0;
  //     for (iage=1; iage<=(nages-3); iage++)
  //     {
  //       sel_MRFSS_D(iyear,iage)=(sel_HB(endyr_period2,iage+3)-sel_HB(endyr_period2,iage));
  //     }
  //     sel_MRFSS_D(iyear,(nages-2))=0.0;
  //     sel_MRFSS_D(iyear,(nages-1))=0.0;
  //     sel_MRFSS_D(iyear,nages)=0.0;
  //   }
  //   else
  //   {
  //     for (iage=1; iage<=(nages-2); iage++)
  //     {
  //       sel_HB_D(iyear,iage)=(sel_HB(endyr,iage+2)-sel_HB(endyr,iage));
  //     }
  //     sel_HB_D(iyear,(nages-1))=0.0;
  //     sel_HB_D(iyear,nages)=0.0;
  //     for (iage=1; iage<=(nages-3); iage++)
  //     {
  //       sel_MRFSS_D(iyear,iage)=(sel_HB(endyr,iage+3)-sel_HB(endyr,iage));
  //     }
  //     sel_MRFSS_D(iyear,(nages-2))=0.0;
  //     sel_MRFSS_D(iyear,(nages-1))=0.0;
  //     sel_MRFSS_D(iyear,nages)=0.0;
  //   }
  //}


  //for (iyear=styr_commHAL_D;iyear<=endyr_commHAL_D;iyear++)
  //{
  //   for (iage=1; iage<=nages; iage++)
  //   {
  //     sel_commHAL_D(iyear,iage)=(max(column(sel_commHAL, iage))-sel_commHAL(endyr,iage));
  //   }
  //   sel_commHAL_D(iyear)=sel_commHAL_D(iyear)/(max(sel_commHAL_D(iyear))+dzero_dum); //prevent division by zero
  //}

  //Alternate way of expressing commercial discard selectivity
  //Uses a 2 age shift
  for (iyear=styr_commHAL_D;iyear<=endyr_commHAL_D;iyear++)
  {
     for (iage=1; iage<=(nages-2); iage++)
     {
       sel_commHAL_D(iyear,iage)=(sel_commHAL(endyr,iage+2)-sel_commHAL(endyr,iage));
     }
     sel_commHAL_D(iyear,(nages-1))=0.0;
     sel_commHAL_D(iyear,nages)=0.0;
  }

FUNCTION get_mortality
 fullF=0.0;
 //initialization F is avg of first 3 yrs (1962-1964)
 log_F_init_commHAL=sum(log_F_dev_commHAL(styr_commHAL_L,(styr_commHAL_L+2)))/3.0;
 log_F_init_commDV=sum(log_F_dev_commDV(styr_commDV_L,(styr_commDV_L+2)))/3.0;
 log_F_init_HB=sum(log_F_dev_HB(styr_HB_L,(styr_HB_L+2)))/3.0;
 log_F_init_MRFSS=sum(log_F_dev_MRFSS(styr_MRFSS_L,(styr_MRFSS_L+2)))/3.0;

 for (iyear=styrR; iyear<=endyr; iyear++)
 {
   if(iyear<styr_commHAL_L)
   {
     F_commHAL_out(iyear)=mfexp(log_avg_F_commHAL+log_F_init_commHAL);
   }
   else
   {
     F_commHAL_out(iyear)=mfexp(log_avg_F_commHAL+log_F_dev_commHAL(iyear));
   }
   F_commHAL(iyear)=sel_commHAL(iyear)*F_commHAL_out(iyear);
   F_commHAL_D(iyear)=sel_commHAL_D(iyear)*Dmort_commHAL*F_commHAL_out(iyear);
   fullF(iyear)+=F_commHAL_out(iyear);

   if(iyear<styr_commDV_L)
   {
     F_commDV_out(iyear)=mfexp(log_avg_F_commDV+log_F_init_commDV);
   }
   else
   {
     F_commDV_out(iyear)=mfexp(log_avg_F_commDV+log_F_dev_commDV(iyear));
   }
   F_commDV(iyear)=sel_commDV(iyear)*F_commDV_out(iyear);
   fullF(iyear)+=F_commDV_out(iyear);

   if(iyear<styr_HB_L)
   {
     F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_init_HB);
   }
   else
   {
     F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_dev_HB(iyear));
   }
   F_HB(iyear)=sel_HB(iyear)*F_HB_out(iyear);
   F_HB_D(iyear)=sel_HB_D(iyear)*Dmort_HB*F_HB_out(iyear);
   fullF(iyear)+=F_HB_out(iyear);

   if(iyear<styr_MRFSS_L)
   {
     F_MRFSS_out(iyear)=mfexp(log_avg_F_MRFSS+log_F_init_MRFSS);
   }
   else
```

```
    {
      F_MRFSS_out(iyear)=mfexp(log_avg_F_MRFSS+log_F_dev_MRFSS(iyear));
    }
    F_MRFSS(iyear)=sel_MRFSS(iyear)*F_MRFSS_out(iyear);
    F_MRFSS_D(iyear)=sel_HB_D(iyear)*Dmort_MRFSS*F_MRFSS_out(iyear); //use HB selectivity
    fullF(iyear)+=F_MRFSS_out(iyear);



    //discards
    if(iyear>=styr_commHAL_D)
    {
      F_commHAL_D_out(iyear)=mfexp(log_avg_F_commHAL_D+log_F_dev_commHAL_D(iyear));
      F_commHAL_D(iyear)=sel_commHAL_D(iyear)*F_commHAL_D_out(iyear);
      fullF(iyear)+=F_commHAL_D_out(iyear);
    }
    if(iyear>=styr_HB_D)
    {
      F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_dev_HB_D(iyear));
      F_HB_D(iyear)=sel_HB_D(iyear)*F_HB_D_out(iyear);
      fullF(iyear)+=F_HB_D_out(iyear);
    }
    if(iyear>=styr_MRFSS_D)
    {
      F_MRFSS_D_out(iyear)=mfexp(log_avg_F_MRFSS_D+log_F_dev_MRFSS_D(iyear));
      F_MRFSS_D(iyear)=sel_MRFSS_D(iyear)*F_MRFSS_D_out(iyear);
      fullF(iyear)+=F_MRFSS_D_out(iyear);
    }

    F(iyear)=F_commHAL(iyear); //first in additive series (NO +=)
    F(iyear)+=F_commDV(iyear);
    F(iyear)+=F_HB(iyear);
    F(iyear)+=F_MRFSS(iyear);

    F(iyear)+=F_commHAL_D(iyear);
    F(iyear)+=F_HB_D(iyear);
    F(iyear)+=F_MRFSS_D(iyear);

    Z(iyear)=M+F(iyear);
  }

FUNCTION get_numbers_at_age
//Initial age
  S0=spr_F0(styrR)*R0;
  B0=bpr_F0(styrR)*R0;
  S1=S0*S1dS0;
  R1=R1_mult*mfexp(log(((0.8*R0*steep*S1)/
     (0.2*R0*spr_F0(styrR)*(1.0-steep)+(steep-0.2)*S1))+dzero_dum));
  N(styrR,1)=R1;
  for (iage=2; iage<=nages; iage++)
  {
    N(styrR,iage)=N(styrR,iage-1)*mfexp(-1.*Z(styrR,iage-1));
  }
  //plus group calculation
  N(styrR,nages)=N(styrR,nages-1)*mfexp(-1.*Z(styrR,nages-1))/
                 (1.-mfexp(-1.*Z(styrR,nages)));
  SSB(styrR)=sum(elem_prod(N(styrR),reprod(styrR)));
  B(styrR)=elem_prod(N(styrR),wgt);
  totB(styrR)=sum(B(styrR));

//Rest of years ages
  for (iyear=styrR; iyear<endyr; iyear++)
  {
    //add 0.00001 to avoid log(zero)
    N(iyear+1,1)=mfexp(log(((0.8*R0*steep*SSB(iyear))/(0.2*R0*spr_F0(iyear)*
        (1.0-steep)+(steep-0.2)*SSB(iyear)))+dzero_dum)+log_dev_N_rec(iyear+1));
    N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
    N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages));//plus group
    SSB(iyear+1)=sum(elem_prod(N(iyear+1),reprod(iyear+1)));
    B(iyear+1)=elem_prod(N(iyear+1),wgt);
    totB(iyear+1)=sum(B(iyear+1));
  }

    //last year (projection) has no recruitment variability
    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_dum));
    N(endyr+1)(2,nages)=++elem_prod(N(endyr)(1,nages-1),(mfexp(-1.*Z(endyr)(1,nages-1))));
    N(endyr+1,nages)+=N(endyr,nages)*mfexp(-1.*Z(endyr,nages));//plus group
    SSB(endyr+1)=sum(elem_prod(N(endyr+1),reprod(endyr)));
    B(endyr+1)=elem_prod(N(endyr+1),wgt);
    totB(endyr+1)=sum(B(endyr+1));

//Recruitment time series
  rec=column(N,1);

//Benchmark parameters
  S1S0=SSB(styr)/S0;
  popstatus=SSB(endyr+1)/S0;


FUNCTION get_catch //Baranov catch eqn
  for (iyear=styrR; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      C_commHAL(iyear,iage)=N(iyear,iage)*F_commHAL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      C_commDV(iyear,iage)=N(iyear,iage)*F_commDV(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      C_HB(iyear,iage)=N(iyear,iage)*F_HB(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      C_MRFSS(iyear,iage)=N(iyear,iage)*F_MRFSS(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
  }

  //pred recreational catches in 1000s
  for (iyear=styr_HB_L; iyear<=endyr_HB_L; iyear++)
    {
     pred_HB_L(iyear)=sum(C_HB(iyear))/1000.0;
    }
  for (iyear=styr_MRFSS_L; iyear<=endyr_MRFSS_L; iyear++)
    {
     pred_MRFSS_L(iyear)=sum(C_MRFSS(iyear))/1000.0;
    }

FUNCTION get_landings

//---Predicted landings------------------------
  for (iyear=styrR; iyear<=endyr; iyear++)
  {
    L_commHAL(iyear)=elem_prod(C_commHAL(iyear),wgt);
    L_commDV(iyear)=elem_prod(C_commDV(iyear),wgt);
    L_HB(iyear)=elem_prod(C_HB(iyear),wgt);
    L_MRFSS(iyear)=elem_prod(C_MRFSS(iyear),wgt);
  }

  for (iyear=styr_commHAL_L; iyear<=endyr_commHAL_L; iyear++)
    {
     pred_commHAL_L(iyear)=sum(L_commHAL(iyear));
    }
  for (iyear=styr_commDV_L; iyear<=endyr_commDV_L; iyear++)
    {
     pred_commDV_L(iyear)=sum(L_commDV(iyear));
    }


FUNCTION get_discards //Baranov catch eqn
  //dead discards at age (number fish)
  for (iyear=styr_commHAL_D; iyear<=endyr_commHAL_D; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      C_commHAL_D(iyear,iage)=N(iyear,iage)*F_commHAL_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_commHAL_D(iyear)=sum(C_commHAL_D(iyear))/1000.0; //pred annual dead discards in 1000s
  }

  for (iyear=styr_HB_D; iyear<=endyr_HB_D; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      C_HB_D(iyear,iage)=N(iyear,iage)*F_HB_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_HB_D(iyear)=sum(C_HB_D(iyear))/1000.0; //pred annual dead discards in 1000s
  }

  for (iyear=styr_MRFSS_D; iyear<=endyr_MRFSS_D; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      C_MRFSS_D(iyear,iage)=N(iyear,iage)*F_MRFSS_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_MRFSS_D(iyear)=sum(C_MRFSS_D(iyear))/1000.0; //pred annual dead discards in 1000s
  }



FUNCTION get_indices
//---Predicted CPUEs------------------------
 //Hook and line Logbook cpue
  for (iyear=styr_HAL_cpue; iyear<=endyr_HAL_cpue; iyear++)
  {
      N_HAL(iyear)=elem_prod(elem_prod(N(iyear),sel_commHAL(iyear)),wgt); //index in weight units
      pred_HAL_cpue(iyear)=mfexp(log_q_HAL)*(1+(iyear-styr_HAL_cpue)*q_rate)*sum(N_HAL(iyear));
      //pred_HAL_cpue(iyear)=mfexp(log_q_HAL)*sum(N_HAL(iyear));
  }
 //Headboat cpue
  for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
  {
      N_HB(iyear)=elem_prod(N(iyear),sel_HB(iyear)); //index in number units
      pred_HB_cpue(iyear)=mfexp(log_q_HB)*(1+(iyear-styr_HB_cpue)*q_rate)*sum(N_HB(iyear));
      //pred_HB_cpue(iyear)=mfexp(log_q_HB)*sum(N_HB(iyear));
  }
  //MRFSS cpue
  for (iyear=styr_MRFSS_cpue; iyear<=endyr_MRFSS_cpue; iyear++)
  {
      N_MRFSS(iyear)=elem_prod(N(iyear),sel_MRFSS(iyear)); //index in number units
      pred_MRFSS_cpue(iyear)=mfexp(log_q_MRFSS)*(1+(iyear-styr_MRFSS_cpue)*q_rate)*sum(N_MRFSS(iyear));
      //pred_MRFSS_cpue(iyear)=mfexp(log_q_MRFSS)*sum(N_MRFSS(iyear));
  }


FUNCTION get_length_comps
 //Commercial
  for (iyear=styr_commHAL_lenc;iyear<=endyr_commHAL_lenc;iyear++)
  {
    pred_commHAL_lenc(iyear)=(C_commHAL(iyear)*lenprob10)/sum(C_commHAL(iyear));
  }
  for (iyear=1;iyear<=nyr_commDV_lenc;iyear++)
  {
    pred_commDV_lenc(iyear)=(C_commDV(yrs_commDV_lenc(iyear))*lenprob10)
                            /sum(C_commDV(yrs_commDV_lenc(iyear)));
  }
 //Headboat
  for (iyear=styr_HB_lenc;iyear<=endyr_HB_lenc;iyear++)
  {
    pred_HB_lenc(iyear)=(C_HB(iyear)*lenprob10)/sum(C_HB(iyear));
  }


FUNCTION get_age_comps
 //Commercial
  for (iyear=1;iyear<=nyr_commHAL_agec;iyear++)
  {
    pred_commHAL_agec(iyear)=C_commHAL(yrs_commHAL_agec(iyear))/
                            sum(C_commHAL(yrs_commHAL_agec(iyear)));
  }
  for (iyear=1;iyear<=nyr_commDV_agec;iyear++)
  {
    pred_commDV_agec(iyear)=C_commDV(yrs_commDV_agec(iyear))/
                            sum(C_commDV(yrs_commDV_agec(iyear)));
  }
 //Headboat
  for (iyear=1;iyear<=nyr_HB_agec;iyear++)
  {
    pred_HB_agec(iyear)=C_HB(yrs_HB_agec(iyear))/sum(C_HB(yrs_HB_agec(iyear)));
  }


//-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_sel_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((3.0*log_avg_F_commHAL+sum(log_F_dev_commHAL(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_commDV+sum(log_F_dev_commDV(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_HB+sum(log_F_dev_HB(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_MRFSS+sum(log_F_dev_MRFSS(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_commHAL_D+sum(log_F_dev_commHAL_D(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_HB_D+sum(log_F_dev_HB_D(endyr-2,endyr)))/3);
  F_temp_sum+=mfexp((3.0*log_avg_F_MRFSS_D+sum(log_F_dev_MRFSS_D(endyr-2,endyr)))/3);

  F_commHAL_prop=mfexp((3.0*log_avg_F_commHAL+sum(log_F_dev_commHAL(endyr-2,endyr)))/3)/F_temp_sum;
  F_commDV_prop=mfexp((3.0*log_avg_F_commDV+sum(log_F_dev_commDV(endyr-2,endyr)))/3)/F_temp_sum;
  F_HB_prop=mfexp((3.0*log_avg_F_HB+sum(log_F_dev_HB(endyr-2,endyr)))/3)/F_temp_sum;
  F_MRFSS_prop=mfexp((3.0*log_avg_F_MRFSS+sum(log_F_dev_MRFSS(endyr-2,endyr)))/3)/F_temp_sum;
  F_commHAL_D_prop=mfexp((3.0*log_avg_F_commHAL_D+sum(log_F_dev_commHAL_D(endyr-2,endyr)))/3)/F_temp_sum;
  F_HB_D_prop=mfexp((3.0*log_avg_F_HB_D+sum(log_F_dev_HB_D(endyr-2,endyr)))/3)/F_temp_sum;
  F_MRFSS_D_prop=mfexp((3.0*log_avg_F_MRFSS_D+sum(log_F_dev_MRFSS_D(endyr-2,endyr)))/3)/F_temp_sum;

  sel_wgted_L=F_commHAL_prop*sel_commHAL(endyr)+
            F_commDV_prop*sel_commDV(endyr)+
            F_HB_prop*sel_HB(endyr)+
            F_MRFSS_prop*sel_MRFSS(endyr);

  sel_wgted_D=F_commHAL_D_prop*sel_commHAL_D(endyr)+
              F_HB_D_prop*sel_HB_D(endyr)+
              F_MRFSS_D_prop*sel_MRFSS_D(endyr);

  sel_wgted_tot=sel_wgted_L+sel_wgted_D;

  max_sel_wgted_tot=max(sel_wgted_tot);
  sel_wgted_tot/=max_sel_wgted_tot;
  sel_wgted_L/=max_sel_wgted_tot; //landings sel bumped up by same amount as total sel
  sel_wgted_D/=max_sel_wgted_tot;



FUNCTION get_msy
  var_rec_dev=norm2(log_dev_N_rec(styr,(endyr-3))-sum(log_dev_N_rec(styr,(endyr-3)))
            /(nyrs-3))/(nyrs-4.); //sample variance yrs 1972-2001
  if (set_BiasCor <= 0.0) {BiasCor=mfexp(var_rec_dev/2.0);}   //bias correction
  else {BiasCor=set_BiasCor;}

  //fill in Fs for per-recruit stuff
  F_msy.fill_seqadd(0,.001);


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

    F_L_age_msy=F_msy(ff)*sel_wgted_L;

    F_D_age_msy=F_msy(ff)*sel_wgted_D;

    Z_age_msy=M+F_L_age_msy+F_D_age_msy;


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


    spr_msy(ff)=sum(elem_prod(N_age_msy,reprod(endyr)));



    //Compute equilibrium values of R (including bias correction), SSB and Yield at each F
    R_eq(ff)=(R0/((5.0*steep-1.0)*spr_msy(ff)))*
              (BiasCor*4.0*steep*spr_msy(ff)-spr_F0(endyr)*(1.0-steep));
    if (R_eq(ff)<dzero_dum) {R_eq(ff)=dzero_dum;}
    N_age_msy*=R_eq(ff);

    for (iage=1; iage<=nages; iage++)
    {
      C_age_msy(iage)=N_age_msy(iage)*(F_L_age_msy(iage)/Z_age_msy(iage))*
                   (1.-mfexp(-1.*Z_age_msy(iage)));
      D_age_msy(iage)=N_age_msy(iage)*(F_D_age_msy(iage)/Z_age_msy(iage))*
                   (1.-mfexp(-1.0*Z_age_msy(iage)));
    }


    SSB_eq(ff)=sum(elem_prod(N_age_msy,reprod(endyr)));
    B_eq(ff)=sum(elem_prod(N_age_msy,wgt));
    L_eq(ff)=sum(elem_prod(C_age_msy,wgt));
    E_eq(ff)=sum(C_age_msy(E_age_st,nages))/sum(N_age_msy(E_age_st,nages));
    D_eq(ff)=sum(D_age_msy)/1000.0;
  }

  msy_out=max(L_eq);

  for(ff=1; ff<=n_iter_msy; ff++)
  {
   if(L_eq(ff) == msy_out)
      {
        SSB_msy_out=SSB_eq(ff);
        B_msy_out=B_eq(ff);
        R_msy_out=R_eq(ff);
        D_msy_out=D_eq(ff);
        E_msy_out=E_eq(ff);
        F_msy_out=F_msy(ff);
        spr_msy_out=spr_msy(ff);
      }
  }

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

  //compute total catch-at-age and landings
  C_total=(C_HB+C_MRFSS+C_commHAL+C_commDV)/1000.0; //catch in 1000s
  L_total=L_HB+L_MRFSS+L_commHAL+L_commDV;

  //compute exploitation rate of age 2+
  for(iyear=styrR; iyear<=endyr; iyear++)
  {
    E(iyear)=(1000.0*sum(C_total(iyear)(E_age_st,nages)))/sum(N(iyear)(E_age_st,nages)); //catch in 1000s
    L_total_yr(iyear)=sum(L_total(iyear));
  }

  steep_sd=steep;
  fullF_sd=fullF;
  E_sd=E;

  if(E_msy_out>0)
    {
      EdE_msy=E/E_msy_out;
      EdE_msy_end=EdE_msy(endyr);
    }
  if(F_msy_out>0)
    {
      FdF_msy=fullF/F_msy_out;
      FdF_msy_end=FdF_msy(endyr);
    }
  if(SSB_msy_out>0)
```

```
    {
      SdSSB_msy=SSB/SSB_msy_out;
      SdSSB_msy_end=SdSSB_msy(endyr+1);
    }


//----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_per_recruit_stuff
  //static per-recruit stuff

  for(iyear=styrR; 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)*mfexp(-1.*Z(iyear,nages-1))/
                   (1.0-mfexp(-1.*Z(iyear,nages)));
    spr_static(iyear)=sum(elem_prod(N_age_spr,reprod(iyear)))/spr_F0(iyear);
  }

  //fill in Fs for per-recruit stuff
  F_spr.fill_seqadd(0,.01);
  //compute SSB/R and YPR as functions of F
  for(int ff=1; ff<=n_iter_spr; ff++)
  {
    //uses fishery-weighted F's, same as in MSY calculations
    Z_age_spr=0.0;
    F_L_age_spr=0.0;

    F_L_age_spr=F_spr(ff)*sel_wgted_L;

    Z_age_spr=M+F_L_age_spr+F_spr(ff)*sel_wgted_D;

    N_age_spr(1)=1.0;
    for (iage=2; iage<=nages; iage++)
    {
      N_age_spr(iage)=N_age_spr(iage-1)*mfexp(-1.*Z_age_spr(iage-1));
    }
    N_age_spr(nages)=N_age_spr(nages-1)*mfexp(-1.*Z_age_spr(nages-1))/
                    (1-mfexp(-1.*Z_age_spr(nages)));
    spr_spr(ff)=sum(elem_prod(N_age_spr,reprod(endyr)));
    L_spr(ff)=0.0;
    for (iage=1; iage<=nages; iage++)
    {
      C_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)+=C_age_spr(iage)*wgt(iage);
    }
    E_spr(ff)=sum(C_age_spr(E_age_st,nages))/sum(N_age_spr(E_age_st,nages));

  }

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


FUNCTION evaluate_objective_function
  fval=0.0;
  fval_unwgt=0.0;

//---likelihoods---------------------------
  f_HAL_cpue=0.0;
  for (iyear=styr_HAL_cpue; iyear<=endyr_HAL_cpue; iyear++)
  {
    f_HAL_cpue+=square(log((pred_HAL_cpue(iyear)+dzero_dum)/
       (obs_HAL_cpue(iyear)+dzero_dum)))/(2.0*square(HAL_cpue_cv(iyear)));
  }
  fval+=w_I_HAL*f_HAL_cpue;
  fval_unwgt+=f_HAL_cpue;

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

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


  f_commHAL_L=0.0; //in 1000s gutted pounds
  for (iyear=styr_commHAL_L; iyear<=endyr_commHAL_L; iyear++)
  {
    if(iyear<=1983)
    {
      f_commHAL_L+=square(log((pred_commHAL_L(iyear)+dzero_dum)/
       (obs_commHAL_L(iyear)*L_commHAL_bias+dzero_dum)))/(2.0*square(commHAL_L_cv(iyear)));
    }
    else
    {
      f_commHAL_L+=square(log((pred_commHAL_L(iyear)+dzero_dum)/
       (obs_commHAL_L(iyear)+dzero_dum)))/(2.0*square(commHAL_L_cv(iyear)));
    }
  }
  fval+=w_L*f_commHAL_L;
  fval_unwgt+=f_commHAL_L;

  f_commDV_L=0.0; //in 1000s gutted pounds
  for (iyear=styr_commDV_L; iyear<=endyr_commDV_L; iyear++)
  {
    f_commDV_L+=square(log((pred_commDV_L(iyear)+dzero_dum)/
       (obs_commDV_L(iyear)+dzero_dum)))/(2.0*square(commDV_L_cv(iyear)));
  }
  fval+=w_L*f_commDV_L;
  fval_unwgt+=f_commDV_L;

  f_HB_L=0.0; //in 1000s
  for (iyear=styr_HB_L; iyear<=endyr_HB_L; iyear++)
  {
    f_HB_L+=square(log((pred_HB_L(iyear)+dzero_dum)/
       (obs_HB_L(iyear)+dzero_dum)))/(2.0*square(HB_L_cv(iyear)));
  }
  fval+=w_L*f_HB_L;
  fval_unwgt+=f_HB_L;

  f_MRFSS_L=0.0; //in 1000s
  for (iyear=styr_MRFSS_L; iyear<=endyr_MRFSS_L; iyear++)
  {
    f_MRFSS_L+=square(log((pred_MRFSS_L(iyear)+dzero_dum)/
       (obs_MRFSS_L(iyear)+dzero_dum)))/(2.0*square(MRFSS_L_cv(iyear)));
  }
  fval+=w_L*f_MRFSS_L;
  fval_unwgt+=f_MRFSS_L;


  f_commHAL_D=0.0; //in 1000s
  for (iyear=styr_commHAL_D; iyear<=endyr_commHAL_D; iyear++)
  {
    f_commHAL_D+=square(log((pred_commHAL_D(iyear)+dzero_dum)/
       (obs_commHAL_D(iyear)+dzero_dum)))/(2.0*square(commHAL_D_cv(iyear)));
  }
  fval+=w_D*f_commHAL_D;
  fval_unwgt+=f_commHAL_D;

  f_HB_D=0.0; //in 1000s
  for (iyear=styr_HB_D; iyear<=endyr_HB_D; iyear++)
  {
    f_HB_D+=square(log((pred_HB_D(iyear)+dzero_dum)/
       (obs_HB_D(iyear)+dzero_dum)))/(2.0*square(HB_D_cv(iyear)));
  }
```

```
  fval+=w_D*f_HB_D;
  fval_unwgt+=f_HB_D;

  f_MRFSS_D=0.0; //in 1000s
  for (iyear=styr_MRFSS_D; iyear<=endyr_MRFSS_D; iyear++)
  {
    f_MRFSS_D+=square(log((pred_MRFSS_D(iyear)+dzero_dum)/
        (obs_MRFSS_D(iyear)+dzero_dum)))/(2.0*square(MRFSS_D_cv(iyear)));
  }
  fval+=w_D*f_MRFSS_D;
  fval_unwgt+=f_MRFSS_D;


  f_commHAL_lenc=0.0;
  for (iyear=styr_commHAL_lenc; iyear<=endyr_commHAL_lenc; iyear++)
  {
    f_commHAL_lenc-=nsamp_commHAL_lenc(iyear)*
        sum(elem_prod((obs_commHAL_lenc(iyear)+dzero_dum),log(pred_commHAL_lenc(iyear)+dzero_dum)));
  }
  fval+=w_lc*f_commHAL_lenc;
  fval_unwgt+=f_commHAL_lenc;

  f_commDV_lenc=0.;
  for (iyear=1; iyear<=nyr_commDV_lenc; iyear++)
  {
    f_commDV_lenc-=nsamp_commDV_lenc(iyear)*
         sum(elem_prod((obs_commDV_lenc(iyear)+dzero_dum),log(pred_commDV_lenc(iyear)+dzero_dum)));
  }
  fval+=w_lc*f_commDV_lenc;
  fval_unwgt+=f_commDV_lenc;

  f_HB_lenc=0.0;
  for (iyear=styr_HB_lenc; iyear<=endyr_HB_lenc; iyear++)
  {
    f_HB_lenc-=nsamp_HB_lenc(iyear)*
        sum(elem_prod((obs_HB_lenc(iyear)+dzero_dum),log(pred_HB_lenc(iyear)+dzero_dum)));
  }
  fval+=w_lc*f_HB_lenc;
  fval_unwgt+=f_HB_lenc;

  f_commHAL_agec=0.0;
  for (iyear=1; iyear<=nyr_commHAL_agec; iyear++)
  {
    f_commHAL_agec-=nsamp_commHAL_agec(iyear)*
          sum(elem_prod((obs_commHAL_agec(iyear)+dzero_dum),log(pred_commHAL_agec(iyear)+dzero_dum)));
  }
  fval+=w_ac*f_commHAL_agec;
  fval_unwgt+=f_commHAL_agec;

  f_commDV_agec=0.0;
  for (iyear=1; iyear<=nyr_commDV_agec; iyear++)
  {
    f_commDV_agec-=nsamp_commDV_agec(iyear)*
          sum(elem_prod((obs_commDV_agec(iyear)+dzero_dum),log(pred_commDV_agec(iyear)+dzero_dum)));
  }
  fval+=w_ac*f_commDV_agec;
  fval_unwgt+=f_commDV_agec;

  f_HB_agec=0.0;
  for (iyear=1; iyear<=nyr_HB_agec; iyear++)
  {
    f_HB_agec-=nsamp_HB_agec(iyear)*
          sum(elem_prod((obs_HB_agec(iyear)+dzero_dum),log(pred_HB_agec(iyear)+dzero_dum)));
  }
  fval+=w_ac*f_HB_agec;
  fval_unwgt+=f_HB_agec;

//-----------Constraints and penalties--------------------------------
  f_N_dev=0.0;
  f_N_dev=norm2(log_dev_N_rec);
  fval+=w_R*f_N_dev;

  f_N_dev_early=0.0;
  f_N_dev_early=norm2(log_dev_N_rec(styrR+1,styr-1));
  fval+=w_R_init*f_N_dev_early;
  f_N_dev_last3=0.0;
    f_N_dev_last3=norm2(log_dev_N_rec(endyr-2,endyr));
  fval+=w_R_end*f_N_dev_last3;

  f_B1dB0_constraint=0.0;
    f_B1dB0_constraint=square(totB(styrR)/B0-B1dB0);
  fval+=w_B1dB0*f_B1dB0_constraint;

  f_Fend_constraint=0.0;
  f_Fend_constraint=norm2(first_difference(fullF(endyr-3,endyr)));
  fval+=w_F*f_Fend_constraint;

  f_fullF_constraint=0.0;
  for (iyear=styrR; iyear<=endyr; iyear++)
    {
     if (fullF(iyear)>5.0)
     {
     f_fullF_constraint+=square(fullF(iyear)-5.0);
     }
    }
  fval+=w_fullF*f_fullF_constraint;

  f_cvlen_diff_constraint=0.0;
    f_cvlen_diff_constraint=norm2(first_difference(log_len_cv_dev));
  fval+=w_cvlen_diff*f_cvlen_diff_constraint;

  f_cvlen_dev_constraint=0.0;
    f_cvlen_dev_constraint=norm2(log_len_cv_dev);
  fval+=w_cvlen_dev*f_cvlen_dev_constraint;

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


REPORT_SECTION
  //cout<<"start report"<<endl;
  get_sel_weighted_current();
  get_msy();
  get_miscellaneous_stuff();
  get_per_recruit_stuff();
  cout << "BC Fmsy=" << F_msy_out<< "   BC SSBmsy=" << SSB_msy_out <<endl;
  cout << "var_rec_resid (72-01)="<<var_rec_dev<<endl;

    report << "TotalLikelihood " << fval << endl;
    report<<" "<<endl;

  report << "Bias-corrected (BC) MSY stuff" << endl;
  report << "BC Fmsy " << F_msy_out << endl;
  report << "BC Emsy(2+) " << E_msy_out << endl;
  report << "BC SSBmsy " << SSB_msy_out << endl;
  report << "BC Rmsy " << R_msy_out << endl;
  report << "BC Bmsy " << B_msy_out << endl;
  report << "BC MSY " << msy_out << endl;
  report << "BC F/Fmsy " << fullF/F_msy_out << endl;
  report << "BC E/Emsy " << E/E_msy_out << endl;
  report << "BC SSB/SSBmsy " << SSB/SSB_msy_out << endl;
  report << "BC B/Bmsy " << totB/B_msy_out << endl;
  report << "BC Yield/MSY " << L_total_yr/msy_out <<endl;
  report << "BC F(2004)/Fmsy " << fullF(endyr)/F_msy_out << endl;
  report << "BC E(2004)/Emsy " << E(endyr)/E_msy_out << endl;
  report << "BC SSB(2005)/SSBmsy " << SSB(endyr+1)/SSB_msy_out << endl;
  report << "BC Predicted Landings(2004)/MSY " << L_total_yr(endyr)/msy_out <<endl;
  report  << " "<<endl;

  report << "Mortality and growth" << endl;
  report << "M "<<M<<endl;
  report << "Linf="<<Linf << "   K=" <<K<<"   t0="<< t0<<endl;
  report << "mean length " << meanlen << endl;
    report << "cv length " << len_cv << endl;
  report << "wgt " << wgt << endl;
  report<<" "<<endl;

  report << "Stock-Recruit " << endl;
  report << "R0= " << R0 << endl;
  report << "Steepness= " << steep << endl;
```

```
 report << "spr_F0= " << spr_F0 << endl;
 report << "Recruits(R) " << rec << endl;
 report << "VirginSSB " << S0 << endl;
 report << "SSB(1978)/VirginSSB " << S1S0 << endl;
 report << "SSB(2004)/VirginSSB " << popstatus << endl;
 report << "SSB " << SSB << endl;
 report << "Biomass " << totB << endl;
 report << "log recruit deviations (1978-2003)   " << log_dev_N_rec(1978,2003) <<endl;
 report << "variance of log rec dev (1978-2000)  "<<var_rec_dev<<endl;
 report<<" "<<endl;

 report << "Exploitation rate (1958-2004)" << endl;
report << E << endl;
report << "Fully-selected F (1958-2004)" << endl;
report << fullF << endl;
report << "Headboat F" << endl;
report << F_HB_out << endl;
report << "MRFSS F" << endl;
report << F_MRFSS_out << endl;
report << "commHAL F" << endl;
report << F_commHAL_out << endl;
report << "commDV F" << endl;
report << F_commDV_out << endl;
 report<<" "<<endl;
report << "Headboat selectivity" << endl;
report << sel_HB << endl;
report << "Headboat DISCARD selectivity" << endl;
report << sel_HB_D << endl;
report << "MRFSS selectivity" << endl;
report << sel_MRFSS << endl;
report << "MRFSS DISCARD selectivity" << endl;
report << sel_MRFSS_D << endl;
report << "commHAL selectivity" << endl;
report << sel_commHAL << endl;
report << "commHAL DISCARD selectivity" << endl;
report << sel_commHAL_D << endl;
report << "commDV selectivity" << endl;
report << sel_commDV << endl;

report << "log_q_HAL "<<log_q_HAL<<endl;
report << "Obs  HAL U"<<obs_HAL_cpue << endl;
report << "pred HAL U"<<pred_HAL_cpue << endl;
report << "log_q_HB "<<log_q_HB<<endl;
report << "Obs  HB U"<<obs_HB_cpue << endl;
report << "pred HB U"<<pred_HB_cpue << endl;
report << "log_q_MRFSS "<<log_q_MRFSS<<endl;
report << "Obs  MRFSS U"<<obs_MRFSS_cpue << endl;
report << "pred MRFSS U"<<pred_MRFSS_cpue << endl;

report << "Obs  HB landings (1000s)"<<obs_HB_L << endl;
report << "pred HB landings (1000s)"<<pred_HB_L << endl;
report << "Obs  MRFSS landings (1000s)"<<obs_MRFSS_L << endl;
report << "pred MRFSS landings (1000s)"<<pred_MRFSS_L << endl;
report << "Obs  commHAL landings (mt)"<<obs_commHAL_L << endl;
report << "pred commHAL landings (mt)"<<pred_commHAL_L << endl;
report << "Obs  commDV landings (mt)"<<obs_commDV_L << endl;
report << "pred commDV landings (mt)"<<pred_commDV_L << endl;

#include "gag_make_Robject2.cxx"   // write the S-compatible report
```

Appendix II. ADMB data file for the gag grouper model.

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  SEDAR Update Assessment: Gag Grouper, May 2006
##
##  Erik Williams, NMFS, Beaufort Lab
##  Erik.Williams@noaa.gov
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#starting and ending year of model
1962
2004
#3 periods of size regs: 1962-91 no restrictions, 1992-98 12inch TL, 1999-04 24inch TL
#ending years of regulation period
1991
1998
##Number of ages (15 classes is 0,1,...10,14+)
21
##vector of agebins, last is a plus group
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
#number length bins
35
#Vector of length bins (mm)(midpoint of bin)
200     230     260     290     320     350     380     410     440     470     500     530     560     590     620     650     680     710     740
        770     800     830     860     890     920     950     980     1010 1040 1070 1100   1130 1160 1190 1220
#discard mortality constant
0.4  #comm HAL
0.25 #headboat
0.25 #MRFSS
#number of iterations in spr calculations (max F examined is (value-1)/100
301
#number of iterations in msy calculations (max F examined is (value-1)/1000
2001
#starting age for exploitation rate: ages are (value-1) to oldest
3
#multiplicative bias correction (may set to 1.0 for none or negative to compute from rec variance)
-1.0
#starting values for VonBert params (Linf, K, t0), units in mm TL
1051
0.24
-0.48
#starting value of constant cv of length at age
0.15
#length-weight coefficients a and b, W=aL^b, (W=kg, L=mm)
3.79956E-08
2.943
#Proportion male by year (1962-2004) (rows) and by age (columns)
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
0.000   0.001   0.003   0.006   0.013   0.026   0.050   0.087   0.141   0.216   0.309   0.416   0.529   0.641   0.742   0.825   0.889   0.934   0.964
        0.981   0.991
```

```
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.003  0.006  0.013  0.026  0.050  0.087  0.141  0.216  0.309  0.416  0.529  0.641  0.742  0.825  0.889  0.934  0.964
       0.981  0.991
0.000  0.001  0.002  0.006  0.012  0.025  0.048  0.086  0.143  0.220  0.318  0.429  0.546  0.658  0.757  0.837  0.897  0.939  0.967
       0.983  0.992
0.000  0.001  0.002  0.005  0.012  0.024  0.047  0.085  0.144  0.225  0.327  0.443  0.563  0.675  0.772  0.849  0.906  0.945  0.970
       0.984  0.993
0.000  0.001  0.002  0.005  0.011  0.023  0.046  0.085  0.145  0.230  0.337  0.457  0.580  0.692  0.787  0.860  0.914  0.950  0.973
       0.986  0.993
0.000  0.001  0.002  0.004  0.010  0.022  0.044  0.084  0.146  0.235  0.346  0.471  0.596  0.709  0.802  0.872  0.922  0.955  0.975
       0.987  0.994
0.000  0.001  0.002  0.004  0.009  0.021  0.043  0.083  0.147  0.239  0.355  0.485  0.613  0.726  0.817  0.884  0.930  0.960  0.978
       0.989  0.995
0.000  0.001  0.001  0.004  0.009  0.020  0.042  0.082  0.149  0.244  0.365  0.499  0.630  0.743  0.832  0.896  0.938  0.965  0.981
       0.991  0.996
0.000  0.000  0.001  0.003  0.008  0.019  0.041  0.082  0.150  0.249  0.374  0.512  0.646  0.760  0.847  0.908  0.946  0.970  0.984
       0.992  0.996
0.000  0.000  0.001  0.003  0.007  0.017  0.039  0.081  0.151  0.253  0.383  0.526  0.663  0.777  0.862  0.919  0.955  0.975  0.987
       0.994  0.997
0.000  0.000  0.001  0.002  0.006  0.016  0.038  0.080  0.152  0.258  0.393  0.540  0.680  0.794  0.877  0.931  0.963  0.980  0.990
       0.995  0.998
0.000  0.000  0.001  0.002  0.006  0.015  0.037  0.080  0.153  0.263  0.402  0.554  0.696  0.812  0.893  0.943  0.971  0.986  0.993
       0.997  0.998
0.000  0.000  0.000  0.002  0.005  0.014  0.035  0.079  0.155  0.268  0.411  0.568  0.713  0.829  0.908  0.955  0.979  0.991  0.996
       0.998  0.999
0.000  0.000  0.000  0.001  0.004  0.013  0.034  0.078  0.156  0.272  0.421  0.582  0.730  0.846  0.923  0.966  0.987  0.996  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.034  0.078  0.156  0.272  0.421  0.582  0.730  0.846  0.923  0.966  0.987  0.996  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.035  0.081  0.162  0.283  0.436  0.598  0.744  0.856  0.929  0.969  0.989  0.996  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.035  0.083  0.168  0.294  0.451  0.615  0.759  0.867  0.935  0.973  0.990  0.997  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.036  0.085  0.174  0.305  0.466  0.631  0.774  0.877  0.942  0.976  0.991  0.997  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.037  0.088  0.180  0.315  0.481  0.648  0.788  0.888  0.948  0.979  0.992  0.998  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.037  0.090  0.186  0.326  0.496  0.664  0.803  0.899  0.954  0.982  0.994  0.998  0.999
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.038  0.093  0.192  0.337  0.511  0.681  0.818  0.909  0.960  0.985  0.995  0.998  1.000
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.038  0.095  0.198  0.348  0.526  0.697  0.832  0.920  0.967  0.988  0.996  0.999  1.000
       1.000  1.000
0.000  0.000  0.000  0.001  0.004  0.013  0.039  0.097  0.204  0.358  0.541  0.714  0.847  0.930  0.973  0.991  0.997  0.999  1.000
       1.000  1.000
0.000  0.000  0.000  0.001  0.003  0.013  0.040  0.100  0.210  0.369  0.556  0.730  0.862  0.941  0.979  0.994  0.999  1.000  1.000
       1.000  1.000
#time-invariant matrix of % maturity-at-age for males
0.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
#time-variant matrix of % maturity-at-age for females
0.000461762 0.005760236 0.067735889 0.476766743 0.919531611 0.993070404 0.999443897 0.999955634 0.999996462 0.999999718 0.999999978 0.999999998 1  1  1  1  1  1  1  1
            1

####################Commercial Hook and Line fishery landings##########################
#Commercial Hook and Line CPUE Index from Logbook
#Starting and ending years of CPUE index
1992
```

```
2004
#Observed CPUE and assumed CVs
0.908   0.944   0.907   0.937   1.001   0.768   0.951   1.017   0.912   0.867   1.006   1.342   1.44
0.0653  0.0422  0.0413  0.041   0.0402  0.0441  0.0432  0.0466  0.0509  0.046   0.0462  0.0465  0.0462
#Commercial Hook and Line fishery landings
#Starting and ending years of landings time series, respectively
1962
2004
#Observed landings (1000 gutted pounds) and assumed CVs
150.3397975 136.9767502 128.3908247 130.4038198 99.11225064 210.9309594 309.9231741 217.1687131 299.0342231 306.7239872 204.4792228 290.4930137 372.7655904 421.7654829 565.0446605 627.5710349 967.3982842 907.5494942 846.1520538
            983.9864299 1027.430745 1101.10127  1108.186651 865.72156   819.8378162 857.7812931 672.3878954 967.0143354 784.2986019 656.4300062 691.6649695 756.6258489 800.0301572 840.428893  751.8975617 608.2184603 654.4595841
            538.0788086 438.2255248 450.0834712 448.334891  443.9049738 476.3863355
0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3         0.3
            0.3         0.3         0.3         0.277272727 0.254545455 0.231818182 0.209090909 0.186363636 0.163636364 0.140909091 0.118181818 0.095454545 0.072727273 0.05        0.05        0.05        0.05        0.05
            0.05        0.05        0.05        0.05        0.05        0.05
#Starting and ending years of discards time series, respectively
1999
2004
#Observed discards (1000s) and assumed CVs
7.369   7.767   13.712  11.91   5.1     7.202
0.1     0.1     0.1     0.1     0.1     0.1
#Starting and ending years of commercial hook and line length composition sample data
1983
2004
#sample size of length comp data by year
116     2856    2689    1449    2125    1368    1241    808     1126    1217    1568    1213    1929    1620    1218    1294    1491    1591    1485
        1034    1161    1621
#commercial length composition samples (year,lengthbin 30mm)
0.000   0.000   0.000   0.000   0.000   0.000   0.003   0.003   0.008   0.010   0.009   0.024   0.034   0.068   0.059   0.032   0.024   0.054   0.078
        0.113   0.072   0.100   0.067   0.060   0.031   0.029   0.029   0.031   0.045   0.011   0.000   0.000   0.009   0.000   0.000
0.000   0.000   0.000   0.000   0.001   0.000   0.002   0.002   0.003   0.009   0.015   0.018   0.019   0.025   0.027   0.028   0.036   0.039   0.047
        0.061   0.070   0.096   0.094   0.079   0.062   0.054   0.050   0.053   0.046   0.033   0.020   0.006   0.004   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.001   0.001   0.001   0.003   0.003   0.002   0.003   0.004   0.007   0.011   0.017   0.017   0.033   0.045
        0.061   0.083   0.089   0.090   0.083   0.086   0.085   0.074   0.073   0.054   0.032   0.027   0.009   0.005   0.000   0.000
0.000   0.000   0.000   0.000   0.003   0.006   0.004   0.007   0.029   0.032   0.047   0.044   0.043   0.041   0.051   0.046   0.060   0.048   0.063
        0.061   0.062   0.073   0.049   0.039   0.041   0.038   0.031   0.033   0.020   0.016   0.009   0.002   0.001   0.000   0.000
0.000   0.000   0.000   0.000   0.001   0.001   0.002   0.003   0.004   0.008   0.011   0.016   0.022   0.029   0.023   0.026   0.037   0.045   0.060
        0.077   0.090   0.088   0.071   0.073   0.058   0.044   0.051   0.037   0.036   0.034   0.022   0.017   0.011   0.001   0.002
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.002   0.003   0.006   0.006   0.003   0.008   0.010   0.023   0.027   0.034   0.065
        0.080   0.092   0.102   0.084   0.070   0.062   0.073   0.069   0.045   0.033   0.041   0.039   0.014   0.006   0.001   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.007   0.007   0.004   0.008   0.017   0.014   0.014   0.027   0.045   0.055   0.062   0.084
        0.091   0.091   0.084   0.072   0.059   0.057   0.055   0.060   0.031   0.024   0.018   0.010   0.004   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.000   0.011   0.011   0.018   0.021   0.022   0.042   0.037   0.058   0.057   0.076   0.089
        0.095   0.090   0.074   0.052   0.061   0.038   0.027   0.030   0.026   0.020   0.018   0.019   0.005   0.001   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.001   0.002   0.003   0.003   0.018   0.018   0.031   0.038   0.051   0.041   0.029   0.048   0.062   0.050
        0.096   0.115   0.101   0.063   0.068   0.039   0.037   0.027   0.019   0.020   0.010   0.005   0.006   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.017   0.059   0.065   0.068   0.079   0.072   0.080   0.084   0.072
        0.090   0.085   0.073   0.041   0.031   0.022   0.011   0.013   0.011   0.010   0.009   0.005   0.001   0.001   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.003   0.032   0.048   0.056   0.065   0.082   0.083   0.085   0.079   0.082
        0.066   0.084   0.051   0.051   0.033   0.022   0.016   0.016   0.016   0.013   0.007   0.005   0.003   0.001   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.022   0.049   0.062   0.080   0.092   0.093   0.105   0.077   0.093
        0.101   0.088   0.037   0.029   0.018   0.014   0.011   0.008   0.009   0.004   0.004   0.002   0.000   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.004   0.023   0.030   0.026   0.041   0.038   0.049   0.053   0.058
        0.057   0.081   0.076   0.082   0.096   0.064   0.049   0.036   0.034   0.039   0.030   0.018   0.011   0.004   0.001   0.000
0.000   0.000   0.000   0.000   0.001   0.000   0.000   0.000   0.000   0.001   0.009   0.027   0.038   0.035   0.042   0.028   0.038   0.047   0.071
        0.083   0.113   0.110   0.092   0.065   0.041   0.040   0.034   0.028   0.022   0.014   0.012   0.003   0.004   0.000   0.001
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.007   0.025   0.035   0.035   0.040   0.051   0.039   0.042   0.062
        0.075   0.086   0.101   0.095   0.075   0.057   0.051   0.029   0.032   0.019   0.021   0.009   0.007   0.004   0.003   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.005   0.017   0.021   0.027   0.022   0.027   0.038   0.052   0.089   0.085
        0.102   0.099   0.072   0.067   0.067   0.047   0.044   0.035   0.024   0.028   0.016   0.010   0.002   0.002   0.001   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.005   0.009   0.021   0.019   0.065   0.129   0.123
        0.135   0.112   0.083   0.061   0.048   0.050   0.038   0.047   0.036   0.010   0.004   0.003   0.000   0.001   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.002   0.001   0.008   0.018   0.023   0.029   0.051   0.088   0.111
        0.139   0.135   0.094   0.074   0.056   0.049   0.040   0.038   0.022   0.016   0.004   0.001   0.000   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.003   0.008   0.018   0.031   0.038   0.038   0.071   0.078
        0.112   0.115   0.095   0.091   0.090   0.061   0.043   0.052   0.030   0.012   0.008   0.004   0.001   0.001   0.001   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.004   0.000   0.008   0.029   0.043   0.045   0.059   0.047   0.069
        0.083   0.091   0.084   0.072   0.078   0.073   0.066   0.055   0.037   0.020   0.011   0.013   0.008   0.000   0.004   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.002   0.007   0.005   0.023   0.023   0.045   0.066   0.107   0.127
        0.119   0.107   0.068   0.065   0.050   0.057   0.039   0.039   0.019   0.013   0.010   0.003   0.000   0.001   0.001   0.002
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.002   0.000   0.004   0.013   0.038   0.034   0.045   0.071   0.081
        0.106   0.110   0.108   0.082   0.060   0.032   0.061   0.074   0.029   0.016   0.017   0.004   0.000   0.005   0.000   0.006
#Number and vector of years of age compositions for hook and line fishery
15
```

```
1979	1980	1981	1992	1993	1994	1995	1996	1997	1999	2000	2001	2002	2003	2004
#sample sizes of age comps by year
144	80	161	66	98	350	816	87	121	88	70	211	139	202	552
#age composition samples (year,age)
0.000	0.000	0.000	0.007	0.063	0.229	0.194	0.125	0.083	0.049	0.069	0.056	0.028	0.000	0.000	0.000	0.007	0.021	0.028
	0.014	0.028
0.000	0.000	0.000	0.013	0.013	0.100	0.238	0.375	0.100	0.075	0.025	0.038	0.013	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.013
0.000	0.000	0.000	0.043	0.143	0.050	0.217	0.161	0.267	0.037	0.031	0.006	0.012	0.019	0.006	0.000	0.000	0.006	0.000
	0.000	0.000
0.000	0.000	0.014	0.135	0.213	0.168	0.165	0.271	0.032	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.017	0.083	0.259	0.217	0.282	0.094	0.047	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
	0.000	0.000
0.000	0.000	0.011	0.070	0.452	0.354	0.075	0.025	0.005	0.004	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.001	0.011	0.037	0.268	0.377	0.132	0.097	0.041	0.016	0.009	0.003	0.002	0.001	0.003	0.001	0.000	0.000
	0.000	0.001
0.000	0.000	0.000	0.000	0.000	0.087	0.582	0.271	0.039	0.003	0.000	0.017	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.000	0.000	0.019	0.021	0.168	0.617	0.122	0.023	0.003	0.024	0.000	0.000	0.003	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.022	0.246	0.106	0.217	0.309	0.062	0.014	0.005	0.000	0.004	0.007	0.000	0.000	0.000	0.009	0.000	0.000
	0.000	0.000
0.000	0.000	0.000	0.103	0.204	0.177	0.347	0.083	0.036	0.013	0.031	0.000	0.000	0.005	0.000	0.000	0.000	0.000	0.000
	0.000	0.001
0.000	0.000	0.003	0.018	0.112	0.182	0.286	0.328	0.025	0.010	0.012	0.014	0.005	0.000	0.001	0.001	0.002	0.000	0.000
	0.000	0.000
0.000	0.000	0.000	0.019	0.082	0.166	0.231	0.299	0.116	0.063	0.007	0.002	0.015	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.001	0.057	0.231	0.364	0.183	0.085	0.044	0.022	0.003	0.004	0.001	0.000	0.004	0.000	0.000	0.000	0.000
	0.000	0.001
0.000	0.000	0.006	0.154	0.299	0.279	0.187	0.032	0.030	0.004	0.003	0.004	0.002	0.001	0.000	0.000	0.000	0.000	0.000
	0.000	0.001

################################Commercial Diving fishery landings##################################################
#Starting and ending years of landings time series, respectively
1976
2004
#Observed landings (1000 gutted pounds)  and CV's
3.752568878 8.809611535 13.86665419 18.92369685 16.40354076 13.88338468 15.8492827 9.07729609 18.74637306 11.61952148 6.341917878 21.93053367 12.96098821 22.25805268 19.06639699 85.01116761 106.7588492 78.15075049 97.50301183
	83.76605227 118.5640778 98.7057244 138.7875457 113.4945167 63.0242389 82.2986539 84.52484584 117.4119668 74.96717967
0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.277272727 0.254545455 0.231818182 0.209090909 0.186363636 0.163636364 0.140909091 0.118181818 0.095454545 0.072727273 0.05
	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
#Number and vector of years of length compositions (comm diving)
4
1999	2000	2001	2003
#sample sizes of length comp data by year
224 198 109 324
#commercial diving length comp samples (year,age) (30mm length bins)
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.003	0.043	0.048	0.021	0.140	0.198
	0.255	0.152	0.027	0.022	0.038	0.000	0.017	0.003	0.014	0.002	0.001	0.011	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.012	0.021	0.020	0.058	0.047	0.080	0.111
	0.158	0.106	0.085	0.045	0.048	0.026	0.024	0.072	0.025	0.033	0.009	0.005	0.008	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.038	0.031	0.038	0.063	0.145
	0.273	0.146	0.058	0.000	0.030	0.031	0.010	0.030	0.056	0.020	0.010	0.010	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.001	0.006	0.008	0.012	0.013	0.012	0.023	0.080	0.141
	0.170	0.178	0.096	0.048	0.029	0.039	0.029	0.048	0.011	0.012	0.019	0.009	0.002	0.004	0.000	0.004
#Number and vector of years of age compositions (comm diving)
3
1994	1995	1997
#sample sizes of age comp data by year
47 63 54
#commercial divign age comp samples (year,age) (30mm length bins)
0.000	0.000	0.085	0.021	0.085	0.234	0.170	0.213	0.149	0.043	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.016	0.016	0.048	0.000	0.460	0.302	0.111	0.032	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000
0.000	0.000	0.019	0.019	0.185	0.056	0.093	0.537	0.093	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000

######################################Headboat landings########################################
#Starting and ending years for CPUE index
1973
```

```
2004
#Observed CPUE values (numbers) and CVs,
2.486       1.762       0.925       0.659       0.678       0.689       1.037       1.198       1.064       1.04        1.15        1.168       0.985       1.006       1.084       1.231       1.166       1.122       1.098
            1.143       1.05        0.872       0.914       0.769       0.821       0.977       0.67        0.713       0.658       0.708       0.522       0.637
0.2737      0.3129      0.442       0.470       0.4647      0.373       0.2887      0.2497      0.2866      0.2592      0.2352      0.2471      0.2506      0.2509      0.2292      0.2059      0.2555      0.253       0.256
            0.2402      0.2726      0.2968      0.2934      0.3635      0.4018      0.2796      0.3826      0.3807      0.3969      0.3903      0.5376      0.4095
#Starting and ending years for landings time series
1962
2004
#Headboat landings vector (1000s fish) and CV's
8.41        7.66        7.18        7.41        5.58        11.77       17.72       12.13       16.66       17.18       13.44       17.99       13.92       8.57        7.56        8.48        6.01        9.55        6.96
            13.86       11.84       16.46       18.69       16.13       17.35       24.09       24.21       22.42       17.59       13.55       13.94       11.8        9.81        10.54       7.5         6.85        8.67
            5.34        5.98        5.12        4.58        3.27        6.66
0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1         0.1
            0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05        0.05
            0.05        0.05        0.05        0.05        0.05        0.05
#Starting and ending years of discards time series, respectively
1981
2004
#Observed discards (1000s) and assumed CVs
0.03        0.02        0.04        0.03        3.76        4.05        5.63        5.65        5.23        4.11        3.16        7.74        6.54        5.45        5.85        4.16        3.81        4.82        4.8
            5.38        4.6         4.12        2.95        6
0.2         0.2         0.2         0.190909091 0.181818182 0.172727273 0.163636364 0.154545455 0.145454545 0.136363636 0.127272727 0.118181818 0.109090909 0.1         0.1         0.1         0.1         0.1         0.1
            0.1         0.1         0.1         0.1         0.1
#Starting and ending year of headboat length composition data
1972
2004
#sample sizes of length comp data by year
150         238         149         240         233         307         182         104         186         465         598         906         1295        991         674         758         612         561         479
            239         366         352         385         558         263         226         410         362         298         291         199         218         199
#recreational length comp samples (year,lengthbin)
0.000       0.000       0.003       0.016       0.013       0.019       0.016       0.019       0.013       0.013       0.032       0.016       0.003       0.044       0.022       0.035       0.032       0.019       0.048
            0.025       0.045       0.041       0.044       0.070       0.073       0.102       0.057       0.022       0.022       0.029       0.045       0.025       0.000       0.038       0.000
0.000       0.000       0.000       0.006       0.003       0.000       0.006       0.011       0.011       0.009       0.022       0.011       0.008       0.014       0.036       0.047       0.025       0.023       0.026
            0.070       0.048       0.053       0.057       0.031       0.090       0.051       0.078       0.078       0.065       0.034       0.040       0.017       0.019       0.000       0.008
0.000       0.004       0.008       0.004       0.008       0.008       0.004       0.004       0.012       0.004       0.012       0.000       0.008       0.004       0.000       0.012       0.015       0.020       0.035
            0.027       0.047       0.058       0.063       0.059       0.059       0.051       0.106       0.138       0.078       0.055       0.087       0.000       0.000       0.012       0.000
0.000       0.000       0.005       0.020       0.021       0.012       0.009       0.012       0.021       0.021       0.021       0.025       0.015       0.014       0.036       0.035       0.046       0.043       0.048
            0.052       0.035       0.058       0.031       0.026       0.029       0.056       0.047       0.089       0.062       0.039       0.036       0.034       0.000       0.005       0.000
0.000       0.004       0.005       0.000       0.008       0.000       0.013       0.008       0.017       0.026       0.034       0.025       0.042       0.030       0.013       0.018       0.029       0.034       0.042
            0.050       0.053       0.068       0.058       0.043       0.042       0.048       0.017       0.046       0.046       0.068       0.066       0.023       0.013       0.004       0.009
0.000       0.005       0.000       0.000       0.005       0.003       0.003       0.003       0.006       0.018       0.020       0.036       0.041       0.043       0.044       0.038       0.074       0.035       0.056
            0.050       0.059       0.083       0.085       0.044       0.011       0.014       0.045       0.039       0.029       0.029       0.033       0.035       0.009       0.000       0.003
0.000       0.000       0.006       0.006       0.000       0.006       0.011       0.006       0.000       0.010       0.011       0.020       0.034       0.034       0.028       0.028       0.048       0.077       0.076
            0.026       0.078       0.070       0.084       0.057       0.037       0.023       0.028       0.028       0.045       0.062       0.028       0.017       0.017       0.000       0.000
0.000       0.000       0.000       0.000       0.000       0.000       0.000       0.000       0.027       0.022       0.000       0.018       0.045       0.045       0.036       0.067       0.063       0.036       0.143
            0.080       0.080       0.085       0.009       0.045       0.018       0.027       0.018       0.058       0.054       0.009       0.009       0.009       0.000       0.000       0.000
0.000       0.000       0.005       0.000       0.000       0.005       0.000       0.005       0.029       0.021       0.037       0.037       0.042       0.021       0.026       0.032       0.084       0.089       0.079
            0.144       0.087       0.076       0.058       0.016       0.000       0.005       0.005       0.031       0.021       0.005       0.021       0.008       0.000       0.010       0.000
0.000       0.000       0.000       0.000       0.007       0.008       0.028       0.009       0.028       0.021       0.056       0.051       0.052       0.059       0.065       0.052       0.037       0.035       0.038
            0.055       0.100       0.102       0.096       0.042       0.027       0.007       0.005       0.013       0.002       0.003       0.000       0.001       0.000       0.000       0.000
0.004       0.000       0.004       0.002       0.004       0.014       0.029       0.029       0.044       0.057       0.040       0.043       0.035       0.019       0.054       0.055       0.053       0.066       0.043
            0.059       0.046       0.083       0.059       0.046       0.029       0.021       0.024       0.014       0.009       0.005       0.003       0.000       0.002       0.002       0.003
0.000       0.000       0.001       0.002       0.003       0.009       0.021       0.051       0.060       0.061       0.072       0.090       0.058       0.062       0.056       0.051       0.039       0.048       0.041
            0.040       0.037       0.056       0.045       0.028       0.024       0.012       0.006       0.010       0.007       0.004       0.002       0.003       0.000       0.000       0.000
0.001       0.000       0.001       0.002       0.004       0.011       0.022       0.024       0.030       0.041       0.035       0.043       0.038       0.048       0.061       0.053       0.071       0.078       0.041
            0.062       0.056       0.086       0.075       0.039       0.021       0.018       0.006       0.014       0.004       0.006       0.003       0.003       0.001       0.001       0.000
0.001       0.006       0.010       0.008       0.009       0.024       0.033       0.027       0.041       0.045       0.053       0.043       0.047       0.065       0.043       0.047       0.045       0.066       0.065
            0.062       0.069       0.064       0.049       0.030       0.020       0.010       0.005       0.001       0.001       0.004       0.003       0.003       0.001       0.000       0.000
0.002       0.011       0.016       0.014       0.023       0.040       0.051       0.067       0.086       0.072       0.065       0.061       0.053       0.055       0.045       0.063       0.050       0.034       0.023
            0.042       0.041       0.032       0.030       0.008       0.002       0.001       0.004       0.004       0.001       0.000       0.002       0.000       0.000       0.000       0.002
0.000       0.007       0.009       0.010       0.013       0.029       0.060       0.069       0.042       0.073       0.052       0.093       0.082       0.073       0.054       0.043       0.051       0.037       0.052
            0.033       0.038       0.026       0.016       0.008       0.013       0.007       0.002       0.006       0.001       0.001       0.002       0.001       0.000       0.000       0.000
0.000       0.006       0.005       0.012       0.012       0.025       0.033       0.033       0.050       0.077       0.070       0.073       0.089       0.055       0.081       0.049       0.060       0.062       0.043
            0.036       0.039       0.041       0.026       0.012       0.002       0.003       0.001       0.001       0.001       0.001       0.001       0.001       0.000       0.000       0.000
0.006       0.021       0.005       0.000       0.017       0.017       0.041       0.058       0.045       0.066       0.063       0.076       0.062       0.071       0.082       0.073       0.058       0.047       0.059
            0.038       0.034       0.034       0.013       0.005       0.006       0.004       0.000       0.002       0.001       0.000       0.000       0.000       0.000       0.000       0.000
0.006       0.009       0.009       0.011       0.023       0.028       0.019       0.055       0.060       0.066       0.062       0.066       0.078       0.058       0.070       0.056       0.060       0.054       0.071
            0.045       0.031       0.027       0.020       0.014       0.003       0.000       0.000       0.000       0.000       0.002       0.000       0.000       0.000       0.000       0.000
0.004       0.022       0.033       0.002       0.000       0.003       0.021       0.031       0.059       0.015       0.057       0.099       0.050       0.083       0.076       0.066       0.053       0.092       0.031
            0.065       0.058       0.025       0.032       0.010       0.006       0.000       0.000       0.002       0.003       0.000       0.000       0.000       0.000       0.000       0.000
0.000       0.000       0.000       0.000       0.000       0.000       0.000       0.003       0.007       0.012       0.047       0.096       0.098       0.116       0.080       0.051       0.095       0.078       0.071
            0.063       0.053       0.054       0.040       0.020       0.011       0.003       0.002       0.000       0.000       0.000       0.000       0.000       0.000       0.000       0.000
```

```
0.000   0.001   0.005   0.000   0.000   0.000   0.004   0.000   0.000   0.003   0.048   0.089   0.074   0.063   0.090   0.060   0.058   0.092   0.073
        0.073   0.062   0.070   0.054   0.041   0.008   0.013   0.003   0.000   0.008   0.000   0.000   0.002   0.004   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.004   0.000   0.001   0.006   0.014   0.026   0.081   0.062   0.068   0.090   0.074   0.077   0.059   0.077
        0.053   0.122   0.066   0.066   0.039   0.003   0.006   0.006   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000   0.002   0.002   0.002   0.000   0.003   0.003   0.003   0.000   0.003   0.041   0.100   0.083   0.067   0.060   0.051   0.048   0.064   0.076
        0.075   0.067   0.098   0.064   0.033   0.015   0.015   0.004   0.000   0.010   0.002   0.006   0.000   0.000   0.001   0.002
0.000   0.001   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.008   0.065   0.117   0.119   0.135   0.096   0.077   0.027   0.036   0.034
        0.047   0.041   0.044   0.056   0.032   0.029   0.022   0.004   0.000   0.000   0.000   0.000   0.000   0.009   0.001   0.000
0.000   0.000   0.000   0.000   0.002   0.000   0.000   0.000   0.009   0.005   0.031   0.107   0.190   0.108   0.100   0.090   0.064   0.052   0.036
        0.043   0.032   0.037   0.027   0.015   0.011   0.005   0.000   0.015   0.000   0.000   0.010   0.000   0.005   0.006   0.000
0.004   0.000   0.000   0.001   0.010   0.000   0.000   0.000   0.002   0.003   0.054   0.106   0.182   0.092   0.123   0.081   0.051   0.056   0.037
        0.066   0.027   0.035   0.042   0.012   0.007   0.006   0.003   0.000   0.002   0.000   0.000   0.000   0.000   0.000   0.000
0.004   0.000   0.000   0.000   0.000   0.000   0.002   0.000   0.003   0.003   0.011   0.035   0.104   0.070   0.196   0.076   0.065   0.083   0.060
        0.067   0.072   0.056   0.041   0.033   0.007   0.000   0.005   0.000   0.000   0.006   0.002   0.000   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.007   0.009   0.000   0.002   0.003   0.005   0.052   0.095   0.162   0.130   0.095   0.062   0.046
        0.049   0.071   0.056   0.054   0.030   0.024   0.002   0.005   0.005   0.004   0.019   0.013   0.000   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.005   0.019   0.036   0.150   0.169   0.075   0.053   0.089
        0.060   0.038   0.090   0.075   0.060   0.014   0.020   0.004   0.011   0.003   0.003   0.004   0.010   0.004   0.010   0.000
0.000   0.000   0.012   0.012   0.000   0.003   0.000   0.006   0.000   0.000   0.005   0.043   0.006   0.003   0.092   0.119   0.066   0.103   0.053
        0.069   0.046   0.085   0.062   0.077   0.039   0.028   0.008   0.001   0.018   0.019   0.016   0.000   0.008   0.000   0.000
0.000   0.000   0.000   0.015   0.000   0.005   0.010   0.010   0.005   0.011   0.000   0.003   0.023   0.026   0.087   0.116   0.107   0.041   0.072
        0.059   0.088   0.079   0.101   0.029   0.035   0.018   0.008   0.013   0.015   0.000   0.001   0.009   0.016   0.000   0.000
0.000   0.000   0.000   0.012   0.000   0.000   0.000   0.000   0.000   0.002   0.000   0.003   0.003   0.055   0.067   0.122   0.103   0.096   0.059
        0.103   0.095   0.077   0.107   0.057   0.033   0.000   0.001   0.000   0.000   0.003   0.000   0.000   0.000   0.000   0.000
#Number and vector of years of age compositions (headboat)
12
1978 1979   1980   1981   1982   1983   1984   1985   1986   2002   2003   2004
#sample sizes of age comp data by year
48 56   80      179     70      283     336     178     140     67      93      85
#headboat age comp samples (year,age)
0.018   0.057   0.258   0.238   0.068   0.084   0.121   0.057   0.057   0.004   0.018   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.018
        0.000   0.000
0.000   0.020   0.305   0.129   0.068   0.207   0.142   0.020   0.081   0.000   0.020   0.000   0.000   0.000   0.000   0.004   0.004   0.000   0.000
        0.000   0.000
0.000   0.060   0.384   0.175   0.025   0.095   0.105   0.090   0.045   0.015   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.005
        0.000   0.000
0.000   0.032   0.222   0.231   0.034   0.039   0.063   0.154   0.170   0.030   0.004   0.008   0.008   0.000   0.008   0.000   0.000   0.000   0.000
        0.000   0.000
0.000   0.047   0.240   0.327   0.076   0.095   0.019   0.107   0.046   0.030   0.000   0.000   0.015   0.000   0.000   0.000   0.000   0.000   0.000
        0.000   0.000
0.000   0.018   0.425   0.117   0.083   0.094   0.109   0.033   0.045   0.017   0.031   0.000   0.015   0.005   0.000   0.003   0.001   0.005   0.000
        0.000   0.000
0.000   0.015   0.035   0.229   0.113   0.093   0.167   0.220   0.026   0.027   0.005   0.043   0.008   0.005   0.004   0.004   0.000   0.004   0.000
        0.000   0.004
0.000   0.141   0.093   0.100   0.323   0.113   0.076   0.107   0.024   0.000   0.000   0.000   0.007   0.000   0.000   0.000   0.000   0.007   0.000
        0.000   0.007
0.000   0.174   0.411   0.096   0.050   0.094   0.069   0.055   0.033   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.008   0.000
        0.000   0.008
0.000   0.000   0.012   0.066   0.044   0.109   0.207   0.342   0.106   0.039   0.033   0.006   0.018   0.006   0.000   0.000   0.000   0.012   0.000
        0.000   0.000
0.000   0.000   0.045   0.050   0.105   0.212   0.124   0.100   0.132   0.085   0.068   0.019   0.012   0.018   0.000   0.000   0.025   0.000   0.000
        0.000   0.006
0.000   0.000   0.084   0.176   0.143   0.211   0.153   0.088   0.046   0.044   0.056   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
        0.000   0.000

###################################MRFSS landings  #################################
#Recreational MRFSS CPUE Index
#Starting and ending years of CPUE index
1981
2004
#Observed CPUE and assumed CVs
0.59    0.487   0.446   0.582   1.294   1.257   1.007   0.615   1.982   0.784   1.018   1.101   1.11    1.036   1.542   0.843   0.877   0.345   1.744
        0.891   0.704   1.196   0.992   1.556
1.2769  1.2314  0.9866  0.880   0.7597  0.6624  0.6722  0.7052  0.6251  0.7028  0.6454  0.6372  0.6406  0.6382  0.6282  0.6537  0.7054  0.7452  0.6373
        0.6516  0.6492  0.6342  0.6494  0.6163
#Recreational Charter+Private boat landings
#Starting and ending years for landings time series
1962
2004
#MRFSS landings vector (1000s fish)
6.17    5.62    5.27    5.44    4.09    8.62    12.98   8.89    12.2    12.59   8.37    12.15   15.68   17.48   23.77   21.94   37.54   35.7    35.39
        56.69   22.17   166.7   165.2   55.31   59.26   97.68   77.08   118.69  63.66   60.9    87.98   83.03   124.51  114.5   86.92   114.74  72.54
        109.31  156.5   90.15   109.76  183.73  135.79
```

```
0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891      0.4891
            0.4651      0.479       0.5232      0.2066      0.2978      0.2693      0.3581      0.2036      0.1749      0.2671      0.1905      0.1353      0.2043      0.172       0.2082      0.2048      0.2186      0.2944
            0.2225      0.1624      0.181       0.1936      0.1573      0.1714
#Starting and ending years of discards time series, respectively
1982
2004
#Observed discards (1000s) and assumed CVs
4.32        91.88       11.95       3.09        12.48       10.3        15.01       43.41       11.46       24.19       38.66       31.23       68.29       73.97       43          82.41       32.22       58.86       126.63
            47.41       85.73       137.62      89.54
0.708700067 1           0.67018881  0.547089864 0.340816901 0.438077355 0.834545008 0.238183165 0.368960844 0.247339455 0.233313137 0.360849774 0.230457347 0.165386745 0.164321037 0.17456048  0.234317987 0.135381757 0.265393085
            0.140370019 0.133114744 0.113872103 0.137439763


################Likelihood Component Weighting##############################################################################
#Weights in objective fcn
50.0        #landings
20.0        #discards
1.0                     #length comps
1.0                     #age comps
5.0                     #HAL cpue index
100.0                   #HB cpue index
1.0           #MRFSS cpue index
1.0                     #S-R residuals
1.0                     #constrain variability of recruitment in initialization period
1.0                     #constrain variability of recruitment in last three yrs
0.0                     #constrain variability of F in last three years
100.0                   #constraint B1/B0
0.0           #penalty if F exceeds 5.0
10.0                    #penalty on deviation in CV at age
10.0        #penalty on first difference in CV at age

################Parameter values and initial guesses###########################################################################
#bias adjustment (multiplier) for commercial landings
1.0
#annual rate of increase on all fishery dependent q's
0.025
#steepness (fixed or initial guess)
0.6
#natural mortality at age
#0.225
0.460       0.271       0.207       0.175       0.156       0.144       0.136       0.131       0.126       0.123       0.121       0.119       0.118       0.117       0.116       0.115       0.115       0.114       0.114
            0.114       0.114
#log catchabilities (initial guesses)
-8.0   #commHAL
-14.0  #HB
-14.0  #MRFSS
#log mean F's (initial guesses)
-2.0                                #commHAL
-8.0                                #commDV
-4.0                                #HB
-3.0                                #MRFSS
#log mean F's for discards (initial guesses)
-5.0                                #commHAL discards
-7.0                                #HB discards
-4.0                                #MRFSS discards
#log_R0 - log virgin recruitment
13.0
#S1dS0 - log recruits in first year         (leave at 1.0)
1.0
#R1_mult: R(styrR)=R1_mult*R1.  R1=R0 if S1dS0=1
1.0
#B1/B0 constraint
1.0
#Selectivity parameters.
#Initial guess should be within boundaries.
3.0 #selpar_L50_commHAL1            b1
2.0 #selpar_slope_commHAL1 b3
4.0 #selpar_L50_commHAL2            b1
2.0 #selpar_slope_commHAL2 b3
5.0 #selpar_L50_commHAL3            b1
3.0 #selpar_slope_commHAL3 b3

3.0 #selpar_L50_commDV1             b1
8.0 #selpar_L502_commDV1            b2
```

```
5.0 #selpar_slope_commDV1                 b3
1.0 #selpar_slope2_commDV1                b4

1.0 #selpar_L50_HB1        b1
5.0 #selpar_slope_HB1      b3
2.0 #selpar_L50_HB2        b1
5.0 #selpar_slope_HB2      b3
3.0 #selpar_L50_HB3        b1
5.0 #selpar_slope_HB3      b3

999 #end of data file flag
```