Beaufort Assessment Model of Southeast US Atlantic snowy grouper (*Epinephelus niveatus* or *Hyporthodus niveatus*): AD Model Builder code and data input file

Sustainable Fisheries Branch, National Marine Fisheries Service, Southeast Fisheries Science Center, Beaufort, NC (contact: Kyle Shertzer)

SEDAR36-WP-15

Submitted: 16 September 2013

Please cite this document as:

Sustainable Fisheries Branch, National Marine Fisheries Service, Southeast Fisheries Science Center, Beaufort, NC. 2013. Beaufort Assessment Model of Southeast US Atlantic snowy grouper (*Epinephelus niveatus* or *Hyporthodus niveatus*): AD Model Builder code and data input file. SEDAR36-WP15. SEDAR, North Charleston, SC. 38 pp.

Notice on SEDAR Working Papers

This information is distributed solely for the purpose of pre-dissemination peer review under applicable information quality guidelines. It has not been formally disseminated by NOAA Fisheries. It does not represent and should not be construed to represent any agency determination or policy.

Beaufort Assessment Model of Southeast US Atlantic snowy grouper (*Epinephelus niveatus* or *Hyporthodus niveatus*): AD Model Builder code and data input file

Sustainable Fisheries Branch, National Marine Fisheries Service,
Southeast Fisheries Science Center,
101 Pivers Island Rd, Beaufort, NC 28516
September 13, 2013

```
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//##
//##  SEDAR 36 SA Snowy Grouper assessment April 2013
//##  NMFS, Beaufort Lab, Sustainable Fisheries Branch
//##
//##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
DATA_SECTION

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

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

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

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

// ending years for selectivity blocks
init_int endyr_selex_phase1;
init_int endyr_selex_phase2;

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

//Total number of ages in population model
init_int nages;
// Vector of ages for age bins in population model
init_vector agebins(1,nages);

//Total number of ages used to match age comps: plus group may differ from popn, first age must not
init_int nages_agec;
//Vector of ages for age bins in age comps
init_vector agebins_agec(1,nages_agec);

//Total number of length bins for each matrix and width of bins)
init_int nlenbins;          //used to match data
init_number lenbins_width;  //width of length bins (mm)

//Vector of lengths for length bins (mm)(midpoint)
init_vector lenbins(1,nlenbins);

//Max F used in spr and msy calcs
init_number max_F_spr_msy;
//Total number of iterations for spr calcs
init_int n_iter_spr;
//Total number of iterations for msy calcs
init_int n_iter_msy;
```

```
//Number years at end of time series over which to average sector F's, for weighted selectivities
init_int selpar_n_yrs_wgted;
//bias correction (set to 1.0 for no bias correction or a negative value to compute from rec variance)
init_number set_BiasCor;

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

//###################MARMAP CVT ############################################################
//CPUE: MMcvt
init_int styr_cvt_cpue;
init_int endyr_cvt_cpue;
init_vector obs_cvt_cpue(styr_cvt_cpue,endyr_cvt_cpue);//Observed CPUE
init_vector cvt_cpue_cv(styr_cvt_cpue,endyr_cvt_cpue); //CV of cpue
// MMcvt Age Compositions
init_int nyr_cvt_agec;
init_ivector yrs_cvt_agec(1,nyr_cvt_agec);
init_vector nsamp_cvt_agec(1,nyr_cvt_agec);
init_vector nfish_cvt_agec(1,nyr_cvt_agec);
init_matrix obs_cvt_agec(1,nyr_cvt_agec,1,nages_agec);


//###################MARMAP vertical longline ###############################################
//CPUE: MMvll
init_int styr_vll_cpue;
init_int endyr_vll_cpue;
init_vector obs_vll_cpue(styr_vll_cpue,endyr_vll_cpue);//Observed CPUE
init_vector vll_cpue_cv(styr_vll_cpue,endyr_vll_cpue); //CV of cpue
// MMvll Age Compositions
init_int nyr_vll_agec;
init_ivector yrs_vll_agec(1,nyr_vll_agec);
init_vector nsamp_vll_agec(1,nyr_vll_agec);
init_vector nfish_vll_agec(1,nyr_vll_agec);
init_matrix obs_vll_agec(1,nyr_vll_agec,1,nages_agec);


//###################Recreational fishery ###################################################
//CPUE: headboat
init_int styr_rec_cpue;
init_int endyr_rec_cpue;
init_vector obs_rec_cpue(styr_rec_cpue,endyr_rec_cpue);//Observed CPUE
init_vector rec_cpue_cv(styr_rec_cpue,endyr_rec_cpue); //CV of cpue

// Landings -- headboat + general rec  (1000s fish)
init_int styr_rec_L;
init_int endyr_rec_L;
init_vector obs_rec_L(styr_rec_L,endyr_rec_L);   //vector of observed landings by year
init_vector rec_L_cv(styr_rec_L,endyr_rec_L);    //vector of CV of landings by year

// Rec (rec) Length Compositions (3 cm bins)
init_int nyr_rec_lenc;
init_ivector yrs_rec_lenc(1,nyr_rec_lenc);
init_vector nsamp_rec_lenc(1,nyr_rec_lenc);
init_vector nfish_rec_lenc(1,nyr_rec_lenc);
init_matrix obs_rec_lenc(1,nyr_rec_lenc,1,nlenbins);

// Rec (rec) Age Compositions
init_int nyr_rec_agec;
init_ivector yrs_rec_agec(1,nyr_rec_agec);
init_vector nsamp_rec_agec(1,nyr_rec_agec);
init_vector nfish_rec_agec(1,nyr_rec_agec);
init_matrix obs_rec_agec(1,nyr_rec_agec,1,nages_agec);

//################Commercial handline fleet ##############################################
//CPUE: cH
init_int styr_cH_cpue;
init_int endyr_cH_cpue;
init_vector obs_cH_cpue(styr_cH_cpue,endyr_cH_cpue); //Observed CPUE
init_vector cH_cpue_cv(styr_cH_cpue,endyr_cH_cpue);  //CV of cpue

// Landings (1000 lb whole weight)
 init_int styr_cH_L;
 init_int endyr_cH_L;
 init_vector obs_cH_L(styr_cH_L,endyr_cH_L);
 init_vector cH_L_cv(styr_cH_L,endyr_cH_L);

// Comm handline length Compositions (3 cm bins)
init_int nyr_cH_lenc;
init_ivector yrs_cH_lenc(1,nyr_cH_lenc);
init_vector nsamp_cH_lenc(1,nyr_cH_lenc);
init_vector nfish_cH_lenc(1,nyr_cH_lenc);
init_matrix obs_cH_lenc(1,nyr_cH_lenc,1,nlenbins);

// Comm handline age compositions
 init_int nyr_cH_agec;
 init_ivector yrs_cH_agec(1,nyr_cH_agec);
 init_vector nsamp_cH_agec(1,nyr_cH_agec);
 init_vector nfish_cH_agec(1,nyr_cH_agec);
 init_matrix obs_cH_agec(1,nyr_cH_agec,1,nages_agec);

//################Commercial longline fleet ##############################################
// Landings (1000 lb whole weight)
```

```
 init_int styr_cL_L;
 init_int endyr_cL_L;
 init_vector obs_cL_L(styr_cL_L,endyr_cL_L);
 init_vector cL_L_cv(styr_cL_L,endyr_cL_L);

// Comm longline length Compositions (3 cm bins)
init_int nyr_cL_lenc;
init_ivector yrs_cL_lenc(1,nyr_cL_lenc);
init_vector nsamp_cL_lenc(1,nyr_cL_lenc);
init_vector nfish_cL_lenc(1,nyr_cL_lenc);
init_matrix obs_cL_lenc(1,nyr_cL_lenc,1,nlenbins);

// Comm longline age compositions
 init_int nyr_cL_agec;
 init_ivector yrs_cL_agec(1,nyr_cL_agec);
 init_vector nsamp_cL_agec(1,nyr_cL_agec);
 init_vector nfish_cL_agec(1,nyr_cL_agec);
 init_matrix obs_cL_agec(1,nyr_cL_agec,1,nages_agec);



//--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><>
//-- BAM DATA_SECTION: parameter section
//--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><--><><

//###################Single Parameter values and initial guesses ################################
// Von Bert parameters in TL mm all fish
init_vector set_Linf(1,7);
init_vector set_K(1,7);
init_vector set_t0(1,7);
//CV of length at age and its standard error all fish
init_vector set_len_cv(1,7);
//Scalar used only for computing MSST. For snowy, this is not M, but fixed at 0.25: MSST=(1-0.25)SSBmsy
init_vector set_M_constant(1,7);
//Spawner-recruit parameters (Initial guesses or fixed values)
init_vector set_steep(1,7);         //recruitment steepness
init_vector set_log_R0(1,7);        //recruitment R0
init_vector set_R_autocorr(1,7);    //recruitment autocorrelation
init_vector set_rec_sigma(1,7);     //recruitment standard deviation in log space
//Initial guesses or fixed values of estimated selectivity parameters
init_vector set_selpar_L50_cvt(1,7);
init_vector set_selpar_slope_cvt(1,7);
init_vector set_selpar_afull_cvt(1,7);
init_vector set_selpar_sigma_cvt(1,7);

init_vector set_selpar_L50_vll(1,7);
init_vector set_selpar_slope_vll(1,7);

init_vector set_selpar_L50_cH(1,7);
init_vector set_selpar_slope_cH(1,7);
//init_vector set_selpar_afull_cH(1,7);
//init_vector set_selpar_sigma_cH(1,7);

init_vector set_selpar_L50_cL(1,7);
init_vector set_selpar_slope_cL(1,7);

init_vector set_selpar_L50_rec(1,7);
init_vector set_selpar_slope_rec(1,7);
init_vector set_selpar_afull_rec(1,7);
init_vector set_selpar_sigma_rec(1,7);

init_vector set_selpar_L50_rec2(1,7);
init_vector set_selpar_slope_rec2(1,7);
init_vector set_selpar_afull_rec2(1,7);
init_vector set_selpar_sigma_rec2(1,7);

init_vector set_selpar_L50_rec3(1,7);
init_vector set_selpar_slope_rec3(1,7);
init_vector set_selpar_afull_rec3(1,7);
init_vector set_selpar_sigma_rec3(1,7);

//--index catchability-----------------------------------------------------------------------------
init_vector set_log_q_cvt(1,7);     //catchability coefficient (log) for MMcvt cpue index
init_vector set_log_q_vll(1,7);     //catchability coefficient (log) MMvll cpue index
init_vector set_log_q_rec(1,7);      //catchability coefficient (log) for rec cpue index
init_vector set_log_q_cH(1,7);      //catchability coefficient (log) for comm handline cpue index

//initial F
init_vector set_F_init(1,7);  //scales initial F
//--mean F's in log space -------------------------------
init_vector set_log_avg_F_cH(1,7);
init_vector set_log_avg_F_cL(1,7);
init_vector set_log_avg_F_rec(1,7);

//###################Dev Vector Parameter values (vals) and bounds ################################
//--F vectors--------------------------
init_vector set_log_F_dev_rec(1,3);
init_vector set_log_F_dev_cH(1,3);
init_vector set_log_F_dev_cL(1,3);
init_vector set_log_rec_dev(1,3);
init_vector set_log_Nage_dev(1,3);
```

```
init_vector set_log_F_dev_rec_vals(styr_rec_L,endyr_rec_L);
init_vector set_log_F_dev_cH_vals(styr_cH_L,endyr_cH_L);
init_vector set_log_F_dev_cL_vals(styr_cL_L,endyr_cL_L);
init_vector set_log_rec_dev_vals(styr_rec_dev,endyr_rec_dev);
init_vector set_log_Nage_dev_vals(2,nages);


//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: likelihood weights section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

init_number set_w_L;            //weight for landings
init_number set_w_I_cvt;        //weight for MM cvt index
init_number set_w_I_vll;        //weight for MM vll index
init_number set_w_I_rec;        //weight for headboat index
init_number set_w_I_cH;         //weight for comm handline index
init_number set_w_lc_rec;       //weight for headboat len comps
init_number set_w_lc_cH;        //weight for comm handline len comps
init_number set_w_lc_cL;        //weight for comm longline len comps
init_number set_w_ac_cvt;       //weight for MM cvt age comps
init_number set_w_ac_vll;       //weight for MM vll age comps
init_number set_w_ac_rec;       //weight for headboat age comps
init_number set_w_ac_cH;        //weight for comm handline age comps
init_number set_w_ac_cL;        //weight for comm longline age comps
init_number set_w_Nage_init;    //for fitting initial abundance at age (excluding first age)
init_number set_w_rec;          //for fitting S-R curve
init_number set_w_rec_early;    //additional constraint on early years recruitment
init_number set_w_rec_end;      //additional constraint on ending years recruitment
init_number set_w_fullF;        //penalty for any Fapex>3(removed in final phase of optimization)
init_number set_w_Ftune;        //weight applied to tuning F (removed in final phase of optimization)

//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
//-- BAM DATA_SECTION: miscellaneous stuff section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

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

//Maturity and proportion female at age
init_vector maturity_f_obs(1,nages);          //proportion females mature at age
init_vector maturity_m_obs(1,nages);          //proportion males mature at age
init_vector prop_f_obs(1,nages);              //proportion female at age

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

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

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

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

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

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

//threshold sample sizes for length comps
init_number minSS_rec_lenc;
init_number minSS_cH_lenc;
init_number minSS_cL_lenc;

//threshold sample sizes for age comps
init_number minSS_cvt_agec;
init_number minSS_vll_agec;
init_number minSS_rec_agec;
init_number minSS_cH_agec;
init_number minSS_cL_agec;

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

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

number sqrt2pi;
number g2mt;                    //conversion of grams to metric tons
```

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

init_number end_of_data_file;
//this section MUST BE INDENTED!!!
 LOCAL_CALCS
  if(end_of_data_file!=999)
  {
     cout << "*** WARNING: Data File NOT READ CORRECTLY ****" << endl;
     exit(0);
  }
  else
  {cout << "Data File read correctly" << endl;}
 END_CALCS


 PARAMETER_SECTION

 LOCAL_CALCS
  const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
  const double K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
  const double t0_LO=set_t0(2); const double t0_HI=set_t0(3); const double t0_PH=set_t0(4);
  const double len_cv_LO=set_len_cv(2); const double len_cv_HI=set_len_cv(3); const double len_cv_PH=set_len_cv(4);
  const double M_constant_LO=set_M_constant(2); const double M_constant_HI=set_M_constant(3); const double M_constant_PH=set_M_constant(4);
  const double steep_LO=set_steep(2); const double steep_HI=set_steep(3); const double steep_PH=set_steep(4);
  const double log_R0_LO=set_log_R0(2); const double log_R0_HI=set_log_R0(3); const double log_R0_PH=set_log_R0(4);
  const double R_autocorr_LO=set_R_autocorr(2); const double R_autocorr_HI=set_R_autocorr(3); const double R_autocorr_PH=set_R_autocorr(4);
  const double rec_sigma_LO=set_rec_sigma(2); const double rec_sigma_HI=set_rec_sigma(3); const double rec_sigma_PH=set_rec_sigma(4);
  const double selpar_L50_cvt_LO=set_selpar_L50_cvt(2); const double selpar_L50_cvt_HI=set_selpar_L50_cvt(3); const double selpar_L50_cvt_PH=set_selpar_L50_cvt(4);
  const double selpar_slope_cvt_LO=set_selpar_slope_cvt(2); const double selpar_slope_cvt_HI=set_selpar_slope_cvt(3); const double selpar_slope_cvt_PH=set_selpar_slope_cvt(4);
  const double selpar_afull_cvt_LO=set_selpar_afull_cvt(2); const double selpar_afull_cvt_HI=set_selpar_afull_cvt(3); const double selpar_afull_cvt_PH=set_selpar_afull_cvt(4);
  const double selpar_sigma_cvt_LO=set_selpar_sigma_cvt(2); const double selpar_sigma_cvt_HI=set_selpar_sigma_cvt(3); const double selpar_sigma_cvt_PH=set_selpar_sigma_cvt(4);
  const double selpar_L50_vll_LO=set_selpar_L50_vll(2); const double selpar_L50_vll_HI=set_selpar_L50_vll(3); const double selpar_L50_vll_PH=set_selpar_L50_vll(4);
  const double selpar_slope_vll_LO=set_selpar_slope_vll(2); const double selpar_slope_vll_HI=set_selpar_slope_vll(3); const double selpar_slope_vll_PH=set_selpar_slope_vll(4);
  const double selpar_L50_cH_LO=set_selpar_L50_cH(2); const double selpar_L50_cH_HI=set_selpar_L50_cH(3); const double selpar_L50_cH_PH=set_selpar_L50_cH(4);
  const double selpar_slope_cH_LO=set_selpar_slope_cH(2); const double selpar_slope_cH_HI=set_selpar_slope_cH(3); const double selpar_slope_cH_PH=set_selpar_slope_cH(4);
  const double selpar_L50_cL_LO=set_selpar_L50_cL(2); const double selpar_L50_cL_HI=set_selpar_L50_cL(3); const double selpar_L50_cL_PH=set_selpar_L50_cL(4);
  const double selpar_slope_cL_LO=set_selpar_slope_cL(2); const double selpar_slope_cL_HI=set_selpar_slope_cL(3); const double selpar_slope_cL_PH=set_selpar_slope_cL(4);
  const double selpar_L50_rec_LO=set_selpar_L50_rec(2); const double selpar_L50_rec_HI=set_selpar_L50_rec(3); const double selpar_L50_rec_PH=set_selpar_L50_rec(4);
  const double selpar_slope_rec_LO=set_selpar_slope_rec(2); const double selpar_slope_rec_HI=set_selpar_slope_rec(3); const double selpar_slope_rec_PH=set_selpar_slope_rec(4);
  const double selpar_afull_rec_LO=set_selpar_afull_rec(2); const double selpar_afull_rec_HI=set_selpar_afull_rec(3); const double selpar_afull_rec_PH=set_selpar_afull_rec(4);
  const double selpar_sigma_rec_LO=set_selpar_sigma_rec(2); const double selpar_sigma_rec_HI=set_selpar_sigma_rec(3); const double selpar_sigma_rec_PH=set_selpar_sigma_rec(4);
  const double selpar_L50_rec2_LO=set_selpar_L50_rec2(2); const double selpar_L50_rec2_HI=set_selpar_L50_rec2(3); const double selpar_L50_rec2_PH=set_selpar_L50_rec2(4);
  const double selpar_slope_rec2_LO=set_selpar_slope_rec2(2); const double selpar_slope_rec2_HI=set_selpar_slope_rec2(3); const double selpar_slope_rec2_PH=set_selpar_slope_rec2(4);
  const double selpar_afull_rec2_LO=set_selpar_afull_rec2(2); const double selpar_afull_rec2_HI=set_selpar_afull_rec2(3); const double selpar_afull_rec2_PH=set_selpar_afull_rec2(4);
  const double selpar_sigma_rec2_LO=set_selpar_sigma_rec2(2); const double selpar_sigma_rec2_HI=set_selpar_sigma_rec2(3); const double
selpar_sigma_rec2_PH=set_selpar_sigma_rec2(4);
  const double selpar_L50_rec3_LO=set_selpar_L50_rec3(2); const double selpar_L50_rec3_HI=set_selpar_L50_rec3(3); const double selpar_L50_rec3_PH=set_selpar_L50_rec3(4);
  const double selpar_slope_rec3_LO=set_selpar_slope_rec3(2); const double selpar_slope_rec3_HI=set_selpar_slope_rec3(3); const double selpar_slope_rec3_PH=set_selpar_slope_rec3(4);
  const double selpar_afull_rec3_LO=set_selpar_afull_rec3(2); const double selpar_afull_rec3_HI=set_selpar_afull_rec3(3); const double selpar_afull_rec3_PH=set_selpar_afull_rec3(4);
  const double selpar_sigma_rec3_LO=set_selpar_sigma_rec3(2); const double selpar_sigma_rec3_HI=set_selpar_sigma_rec3(3); const double
selpar_sigma_rec3_PH=set_selpar_sigma_rec3(4);

  const double log_q_cvt_LO=set_log_q_cvt(2); const double log_q_cvt_HI=set_log_q_cvt(3); const double log_q_cvt_PH=set_log_q_cvt(4);
  const double log_q_vll_LO=set_log_q_vll(2); const double log_q_vll_HI=set_log_q_vll(3); const double log_q_vll_PH=set_log_q_vll(4);
  const double log_q_rec_LO=set_log_q_rec(2); const double log_q_rec_HI=set_log_q_rec(3); const double log_q_rec_PH=set_log_q_rec(4);
  const double log_q_cH_LO=set_log_q_cH(2); const double log_q_cH_HI=set_log_q_cH(3); const double log_q_cH_PH=set_log_q_cH(4);

  const double F_init_LO=set_F_init(2); const double F_init_HI=set_F_init(3); const double F_init_PH=set_F_init(4);
  const double log_avg_F_cH_LO=set_log_avg_F_cH(2); const double log_avg_F_cH_HI=set_log_avg_F_cH(3); const double log_avg_F_cH_PH=set_log_avg_F_cH(4);
  const double log_avg_F_cL_LO=set_log_avg_F_cL(2); const double log_avg_F_cL_HI=set_log_avg_F_cL(3); const double log_avg_F_cL_PH=set_log_avg_F_cL(4);
  const double log_avg_F_rec_LO=set_log_avg_F_rec(2); const double log_avg_F_rec_HI=set_log_avg_F_rec(3); const double log_avg_F_rec_PH=set_log_avg_F_rec(4);

  //-dev vectors-----------------------------------------------------------------------------------------------------
  const double log_F_dev_rec_LO=set_log_F_dev_rec(1); const double log_F_dev_rec_HI=set_log_F_dev_rec(2); const double log_F_dev_rec_PH=set_log_F_dev_rec(3);
  const double log_F_dev_cH_LO=set_log_F_dev_cH(1); const double log_F_dev_cH_HI=set_log_F_dev_cH(2); const double log_F_dev_cH_PH=set_log_F_dev_cH(3);
  const double log_F_dev_cL_LO=set_log_F_dev_cL(1); const double log_F_dev_cL_HI=set_log_F_dev_cL(2); const double log_F_dev_cL_PH=set_log_F_dev_cL(3);
  const double log_rec_dev_LO=set_log_rec_dev(1); const double log_rec_dev_HI=set_log_rec_dev(2); const double log_rec_dev_PH=set_log_rec_dev(3);
  const double log_Nage_dev_LO=set_log_Nage_dev(1); const double log_Nage_dev_HI=set_log_Nage_dev(2); const double log_Nage_dev_PH=set_log_Nage_dev(3);

 END_CALCS

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

  vector meanlen_TL(1,nages);   //mean total length (mm) at age all fish
  vector wgt_g(1,nages);        //whole wgt in g
  vector wgt_kg(1,nages);       //whole wgt in kg
  vector wgt_mt(1,nages);       //whole wgt in mt
  vector wgt_klb(1,nages);      //whole wgt in 1000 lb
  vector wgt_lb(1,nages);       //whole wgt in lb
//  vector gonad_wgt_mt(1,nages); //gonad wgt in mt
```

```
  matrix len_rec_mm(styr,endyr,1,nages);         //mean length at age of rec landings in mm (may differ from popn mean)
  matrix wholewgt_rec_klb(styr,endyr,1,nages);   //whole wgt of rec landings in 1000 lb
  matrix len_cH_mm(styr,endyr,1,nages);          //mean length at age of commercial handline landings in mm (may differ from popn mean)
  matrix wholewgt_cH_klb(styr,endyr,1,nages);    //whole wgt of commercial handline landings in 1000 lb
  matrix len_cL_mm(styr,endyr,1,nages);          //mean length at age of commercial handline landings in mm (may differ from popn mean)
  matrix wholewgt_cL_klb(styr,endyr,1,nages);    //whole wgt of commercial handline landings in 1000 lb

  //vector lbins(1,nlenbins); //NOT NEEDED, USE lenbins instead, after making it a vector instead of ivector
  matrix lenprob(1,nages,1,nlenbins);          //distn of size at age (age-length key, 3 cm bins) in population
  number zscore_len;                          //standardized normal values used for computing lenprob
  vector cprob_lenvec(1,nlenbins);             //cumulative probabilities used for computing lenprob
  number zscore_lzero;                        //standardized normal values for length = 0
  number cprob_lzero;                         //length probability mass below zero, used for computing lenprob


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

  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph

//----Predicted length and age compositions
  matrix pred_rec_lenc(1,nyr_rec_lenc,1,nlenbins);
  matrix pred_cH_lenc(1,nyr_cH_lenc,1,nlenbins);
  matrix pred_cL_lenc(1,nyr_cL_lenc,1,nlenbins);

  matrix pred_cvt_agec(1,nyr_cvt_agec,1,nages_agec);
  matrix pred_cvt_agec_allages(1,nyr_cvt_agec,1,nages);
  matrix ErrorFree_cvt_agec(1,nyr_cvt_agec,1,nages);
  matrix pred_vll_agec(1,nyr_vll_agec,1,nages_agec);
  matrix pred_vll_agec_allages(1,nyr_vll_agec,1,nages);
  matrix ErrorFree_vll_agec(1,nyr_vll_agec,1,nages);
  matrix pred_rec_agec(1,nyr_rec_agec,1,nages_agec);
  matrix pred_rec_agec_allages(1,nyr_rec_agec,1,nages);
  matrix ErrorFree_rec_agec(1,nyr_rec_agec,1,nages);
  matrix pred_cH_agec(1,nyr_cH_agec,1,nages_agec);
  matrix pred_cH_agec_allages(1,nyr_cH_agec,1,nages);
  matrix ErrorFree_cH_agec(1,nyr_cH_agec,1,nages);
  matrix pred_cL_agec(1,nyr_cL_agec,1,nages_agec);
  matrix pred_cL_agec_allages(1,nyr_cL_agec,1,nages);
  matrix ErrorFree_cL_agec(1,nyr_cL_agec,1,nages);

//effective sample size applied in multinomial distributions
  vector nsamp_rec_lenc_allyr(styr,endyr);
  vector nsamp_cH_lenc_allyr(styr,endyr);
  vector nsamp_cL_lenc_allyr(styr,endyr);
  vector nsamp_cvt_agec_allyr(styr,endyr);
  vector nsamp_vll_agec_allyr(styr,endyr);
  vector nsamp_rec_agec_allyr(styr,endyr);
  vector nsamp_cH_agec_allyr(styr,endyr);
  vector nsamp_cL_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_rec_lenc_allyr(styr,endyr);
  vector nfish_cH_lenc_allyr(styr,endyr);
  vector nfish_cL_lenc_allyr(styr,endyr);
  vector nfish_cvt_agec_allyr(styr,endyr);
  vector nfish_vll_agec_allyr(styr,endyr);
  vector nfish_rec_agec_allyr(styr,endyr);
  vector nfish_cH_agec_allyr(styr,endyr);
  vector nfish_cL_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_rec_lenc_allyr_out(styr,endyr);
  vector neff_cH_lenc_allyr_out(styr,endyr);
  vector neff_cL_lenc_allyr_out(styr,endyr);
  vector neff_cvt_agec_allyr_out(styr,endyr);
  vector neff_vll_agec_allyr_out(styr,endyr);
  vector neff_rec_agec_allyr_out(styr,endyr);
  vector neff_cH_agec_allyr_out(styr,endyr);
  vector neff_cL_agec_allyr_out(styr,endyr);

//-----Population----------------------------------------------------------------------------------
  matrix N(styr,endyr+1,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and cpue
  matrix N_spawn(styr,endyr,1,nages);       //Population numbers by year and age at peaking spawning: used for SSB
  init_bounded_vector log_Nage_dev(2,nages,log_Nage_dev_LO,log_Nage_dev_HI,log_Nage_dev_PH);
  vector log_Nage_dev_output(1,nages);      //used in output. equals zero for first age
  matrix B(styr,endyr+1,1,nages);           //Population biomass by year and age at start of yr
  vector totB(styr,endyr+1);                //Total biomass by year
  vector totN(styr,endyr+1);                //Total abundance by year
  vector SSB(styr,endyr);                   //Total spawning biomass by year (female + male mature biomass)
  vector MatFemB(styr,endyr);               //Total spawning biomass by year (mature female biomass)
  vector rec(styr,endyr+1);                 //Recruits by year
  vector prop_f(1,nages);                   //Proportion female by age, all males assumed mature
  vector maturity_f(1,nages);               //Proportion of female mature at age
  vector maturity_m(1,nages);               //Proportion of male mature at age
  vector reprod(1,nages);                   //vector used to compute spawning biomass (total mature biomass - males + females)
  vector reprod2(1,nages);                  //vector used to compute mature female biomass
```

```
//---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------
  init_bounded_number log_R0(log_R0_LO,log_R0_HI,log_R0_PH);       //log(virgin Recruitment)
  vector log_R0_out(1,8);
  number R0;                              //virgin recruitment
  init_bounded_number steep(steep_LO,steep_HI,steep_PH); //steepness
  vector steep_out(1,8);
  init_bounded_number rec_sigma(rec_sigma_LO,rec_sigma_HI,rec_sigma_PH);  //sd recruitment residuals  //KC comment out; this should fix it at the initial value
  vector rec_sigma_out(1,8);
  init_bounded_number R_autocorr(R_autocorr_LO,R_autocorr_HI,R_autocorr_PH);  //autocorrelation in SR  KC commented out since not estimated
  vector R_autocorr_out(1,8);

  number rec_sigma_sq;                    //square of rec_sigma
  number rec_logL_add;                    //additive term in -logL term

  init_bounded_dev_vector log_rec_dev(styr_rec_dev,endyr_rec_dev,log_rec_dev_LO,log_rec_dev_HI,log_rec_dev_PH);
  vector log_rec_dev_output(styr,endyr+1);            //used in t.series output. equals zero except for yrs in log_rec_dev
  vector log_rec_dev_out(styr_rec_dev,endyr_rec_dev);  //used in output for bound checking

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

  number BiasCor;                         //Bias correction in equilibrium recruits

  number S0;                              //equal to spr_F0*R0 = virgin SSB
  number B0;                              //equal to bpr_F0*R0 = virgin B
  number R1;                              //Recruits in styr
  number R_virgin;                        //unfished recruitment with bias correction
  vector SdS0(styr,endyr);                //SSB / virgin SSB


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

//MARMAP cvt-----------------------------------------------
  matrix sel_cvt(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cvt(selpar_L50_cvt_LO,selpar_L50_cvt_HI,selpar_L50_cvt_PH);
  init_bounded_number selpar_slope_cvt(selpar_slope_cvt_LO,selpar_slope_cvt_HI,selpar_slope_cvt_PH);
  init_bounded_number selpar_afull_cvt(selpar_afull_cvt_LO,selpar_afull_cvt_HI,selpar_afull_cvt_PH);
  init_bounded_number selpar_sigma_cvt(selpar_sigma_cvt_LO,selpar_sigma_cvt_HI,selpar_sigma_cvt_PH);
  vector selpar_L50_cvt_out(1,8);
  vector selpar_slope_cvt_out(1,8);
  vector selpar_afull_cvt_out(1,8);
  vector selpar_sigma_cvt_out(1,8);

//MARMAP vll-----------------------------------------------
  matrix sel_vll(styr,endyr,1,nages);
  init_bounded_number selpar_L50_vll(selpar_L50_vll_LO,selpar_L50_vll_HI,selpar_L50_vll_PH);
  init_bounded_number selpar_slope_vll(selpar_slope_vll_LO,selpar_slope_vll_HI,selpar_slope_vll_PH);
  vector selpar_L50_vll_out(1,8);
  vector selpar_slope_vll_out(1,8);

//Commercial handline-----------------------------------------------
  matrix sel_cH(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cH(selpar_L50_cH_LO,selpar_L50_cH_HI,selpar_L50_cH_PH);
  init_bounded_number selpar_slope_cH(selpar_slope_cH_LO,selpar_slope_cH_HI,selpar_slope_cH_PH);
  vector selpar_L50_cH_out(1,8);
  vector selpar_slope_cH_out(1,8);

//Commercial longline-----------------------------------------------
  matrix sel_cL(styr,endyr,1,nages);
  init_bounded_number selpar_L50_cL(selpar_L50_cL_LO,selpar_L50_cL_HI,selpar_L50_cL_PH);
  init_bounded_number selpar_slope_cL(selpar_slope_cL_LO,selpar_slope_cL_HI,selpar_slope_cL_PH);
  vector selpar_L50_cL_out(1,8);
  vector selpar_slope_cL_out(1,8);

//Recreational-----------------------------------------------
  matrix sel_rec(styr,endyr,1,nages);
  init_bounded_number selpar_L50_rec(selpar_L50_rec_LO,selpar_L50_rec_HI,selpar_L50_rec_PH);
  init_bounded_number selpar_slope_rec(selpar_slope_rec_LO,selpar_slope_rec_HI,selpar_slope_rec_PH);
  init_bounded_number selpar_afull_rec(selpar_afull_rec_LO,selpar_afull_rec_HI,selpar_afull_rec_PH);
  init_bounded_number selpar_sigma_rec(selpar_sigma_rec_LO,selpar_sigma_rec_HI,selpar_sigma_rec_PH);
  init_bounded_number selpar_L50_rec2(selpar_L50_rec2_LO,selpar_L50_rec2_HI,selpar_L50_rec2_PH);
  init_bounded_number selpar_slope_rec2(selpar_slope_rec2_LO,selpar_slope_rec2_HI,selpar_slope_rec2_PH);
  init_bounded_number selpar_afull_rec2(selpar_afull_rec2_LO,selpar_afull_rec2_HI,selpar_afull_rec2_PH);
  init_bounded_number selpar_sigma_rec2(selpar_sigma_rec2_LO,selpar_sigma_rec2_HI,selpar_sigma_rec2_PH);
  init_bounded_number selpar_L50_rec3(selpar_L50_rec3_LO,selpar_L50_rec3_HI,selpar_L50_rec3_PH);
  init_bounded_number selpar_slope_rec3(selpar_slope_rec3_LO,selpar_slope_rec3_HI,selpar_slope_rec3_PH);
  init_bounded_number selpar_afull_rec3(selpar_afull_rec3_LO,selpar_afull_rec3_HI,selpar_afull_rec3_PH);
  init_bounded_number selpar_sigma_rec3(selpar_sigma_rec3_LO,selpar_sigma_rec3_HI,selpar_sigma_rec3_PH);

  vector selpar_L50_rec_out(1,8);
  vector selpar_slope_rec_out(1,8);
  vector selpar_afull_rec_out(1,8);
  vector selpar_sigma_rec_out(1,8);
  vector selpar_L50_rec2_out(1,8);
  vector selpar_slope_rec2_out(1,8);
  vector selpar_afull_rec2_out(1,8);
  vector selpar_sigma_rec2_out(1,8);
  vector selpar_L50_rec3_out(1,8);
  vector selpar_slope_rec3_out(1,8);
  vector selpar_afull_rec3_out(1,8);
  vector selpar_sigma_rec3_out(1,8);
```

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

//-------------------------------------------------------------------------------------------------------------------------------------
//-------CPUE Predictions--------------------------------
  vector pred_cvt_cpue(styr_cvt_cpue,endyr_cvt_cpue);              //predicted cvt index (number fish per effort)
  matrix N_cvt(styr_cvt_cpue,endyr_cvt_cpue,1,nages);              //used to compute cvt index
  vector pred_vll_cpue(styr_vll_cpue,endyr_vll_cpue);             //predicted vll index (number fish per effort)
  matrix N_vll(styr_vll_cpue,endyr_vll_cpue,1,nages);             //used to compute vll index
  vector pred_rec_cpue(styr_rec_cpue,endyr_rec_cpue);                //predicted rec index (number fish per effort)
  matrix N_rec(styr_rec_cpue,endyr_rec_cpue,1,nages);                //used to compute rec index
  vector pred_cH_cpue(styr_cH_cpue,endyr_cH_cpue);                   //predicted cH index (weight fish per effort)
  matrix N_cH(styr_cH_cpue,endyr_cH_cpue,1,nages);                   //used to compute cH index


//---Catchability (CPUE q's)-----------------------------------------------------------
  init_bounded_number log_q_cvt(log_q_cvt_LO,log_q_cvt_HI,log_q_cvt_PH);
  init_bounded_number log_q_vll(log_q_vll_LO,log_q_vll_HI,log_q_vll_PH);
  init_bounded_number log_q_rec(log_q_rec_LO,log_q_rec_HI,log_q_rec_PH);
  init_bounded_number log_q_cH(log_q_cH_LO,log_q_cH_HI,log_q_cH_PH);
  vector log_q_cvt_out(1,8);
  vector log_q_vll_out(1,8);
  vector log_q_rec_out(1,8);
  vector log_q_cH_out(1,8);


//  init_bounded_number q_rate(0.001,0.1,set_q_rate_phase);  //KC not estimated so commented out, declared as number
  number q_rate;
  vector q_rate_fcn_rec(styr_rec_cpue,endyr_rec_cpue);         //increase due to technology creep (saturates in 2003) //KWS
  vector q_rate_fcn_cH(styr_cH_cpue,endyr_cH_cpue);         //increase due to technology creep (saturates in 2003) //KWS

//  init_bounded_number q_DD_beta(0.1,0.9,set_q_DD_phase);    //KC not estimated so commented out and declared as number (below)
  number q_DD_beta;
  vector q_DD_fcn(styr,endyr);    //density dependent function as a multiple of q (scaled a la Katsukawa and Matsuda. 2003)
  number B0_q_DD;                 //B0 of ages q_DD_age plus
  vector B_q_DD(styr,endyr);      //annual biomass of ages q_DD_age plus

//Fishery dependent random walk catchability
//  init_bounded_vector q_RW_log_dev_rec(styr_rec_cpue,endyr_rec_cpue-1,-3.0,3.0,set_q_RW_phase); //NOT estimated in this model
  vector q_RW_log_dev_rec(styr_rec_cpue,endyr_rec_cpue-1);
  vector q_RW_log_dev_cH(styr_cH_cpue,endyr_cH_cpue-1);

//Fishery dependent catchability over time, may be constant
  vector q_rec(styr_rec_cpue,endyr_rec_cpue);
  vector q_cH(styr_cH_cpue,endyr_cH_cpue);

//-------------------------------------------------------------------------------------------------------------------------------------
//---Landings in numbers (total or 1000 fish) and in wgt (klb)-------------------------------------------------------
  matrix L_rec_num(styr,endyr,1,nages);  //landings (numbers) at age
  matrix L_rec_klb(styr,endyr,1,nages);  //landings (1000 lb whole weight) at age
  vector pred_rec_L_knum(styr,endyr);    //yearly landings in 1000 fish summed over ages
  vector pred_rec_L_klb(styr,endyr);     //yearly landings in 1000 lb summed over ages

  matrix L_cH_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cH_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cH_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cH_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages
  //vector obs_cH_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicHtive bias

  matrix L_cL_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cL_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cL_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cL_L_klb(styr,endyr);      //yearly landings in 1000 lb summed over ages
  //vector obs_cL_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicLtive bias

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

////---MSY calcs-------------------------------------------------------------------------
  number F_rec_prop;      //proportion of F_sum attributable to rec, last X=selpar_n_yrs_wgted yrs, used for avg body weights
  number F_cH_prop;       //proportion of F_sum attributable to cH, last X yr
  number F_cL_prop;       //proportion of F_sum attributable to cL, last X yr
  number F_init_cH_prop;  //proportion of F_init attributable to cH, first X yrs
  number F_init_rec_prop;  //proportion of F_init attributable to rec, first X yrs

  number F_temp_sum;      //sum of geom mean Fsum's in last X yrs, used to compute F_fishery_prop

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

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

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

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

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

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

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

  vector M(1,nages);                        //age-dependent natural mortality
  init_bounded_number M_constant(M_constant_LO,M_constant_HI,M_constant_PH);                        //age-indpendent: used only for MSST
  vector M_constant_out(1,8);
  number smsy2msst;                         //scales msy to get msst, usually (1-M), here 0.75. Used only in output.

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

  init_bounded_number log_avg_F_rec(log_avg_F_rec_LO,log_avg_F_rec_HI,log_avg_F_rec_PH);
  vector log_avg_F_rec_out(1,8);
  init_bounded_dev_vector log_F_dev_rec(styr_rec_L,endyr_rec_L,log_F_dev_rec_LO,log_F_dev_rec_HI,log_F_dev_rec_PH);
  vector log_F_dev_rec_out(styr_rec_L,endyr_rec_L);
  matrix F_rec(styr,endyr,1,nages);
  vector F_rec_out(styr,endyr); //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_rec;
  number log_F_dev_end_rec;

  init_bounded_number log_avg_F_cH(log_avg_F_cH_LO,log_avg_F_cH_HI,log_avg_F_cH_PH);
  vector log_avg_F_cH_out(1,8);
  init_bounded_dev_vector log_F_dev_cH(styr_cH_L,endyr_cH_L,log_F_dev_cH_LO,log_F_dev_cH_HI,log_F_dev_cH_PH);
  vector log_F_dev_cH_out(styr_cH_L,endyr_cH_L);
  matrix F_cH(styr,endyr,1,nages);
  vector F_cH_out(styr,endyr);       //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_cH;
  number log_F_dev_end_cH;

  init_bounded_number log_avg_F_cL(log_avg_F_cL_LO,log_avg_F_cL_HI,log_avg_F_cL_PH);
  vector log_avg_F_cL_out(1,8);
  init_bounded_dev_vector log_F_dev_cL(styr_cL_L,endyr_cL_L,log_F_dev_cL_LO,log_F_dev_cL_HI,log_F_dev_cL_PH);
  vector log_F_dev_cL_out(styr_cL_L,endyr_cL_L);
  matrix F_cL(styr,endyr,1,nages);
  vector F_cL_out(styr,endyr);       //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_cL;

//  init_bounded_number F_init(F_init_LO,F_init_HI,F_init_PH);
  init_bounded_number F_init(F_init_LO,F_init_HI,F_init_PH); //scales early F for initialization
  vector F_init_out(1,8);

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

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

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

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

  number sdnr_lc_rec;
```

```
  number sdnr_lc_cH;
  number sdnr_lc_cL;

  number sdnr_ac_cvt;
  number sdnr_ac_vll;
  number sdnr_ac_rec;
  number sdnr_ac_cH;
  number sdnr_ac_cL;

  number sdnr_I_cvt;
  number sdnr_I_vll;
  number sdnr_I_rec;
  number sdnr_I_cH;

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

  number w_I_cvt;
  number w_I_vll;
  number w_I_rec;
  number w_I_cH;

  number w_lc_rec;
  number w_lc_cH;
  number w_lc_cL;

  number w_ac_cvt;
  number w_ac_vll;
  number w_ac_rec;
  number w_ac_cH;
  number w_ac_cL;

  number w_Nage_init;
  number w_rec;
  number w_rec_early;
  number w_rec_end;
  number w_fullF;
  number w_Ftune;

  number f_rec_L;
  number f_cH_L;
  number f_cL_L;

  number f_cvt_cpue;
  number f_vll_cpue;
  number f_rec_cpue;
  number f_cH_cpue;

  number f_rec_lenc;
  number f_cH_lenc;
  number f_cL_lenc;

  number f_cvt_agec;
  number f_vll_agec;
  number f_rec_agec;
  number f_cH_agec;
  number f_cL_agec;

  //Penalties and constraints. Not all are used.
  number f_Nage_init;               //weight on log devs to estimate initial abundance (excluding first age)
  number f_rec_dev;                 //weight on recruitment deviations to fit S-R curve
  number f_rec_dev_early;           //extra weight on deviations in first recruitment stanza
  number f_rec_dev_end;             //extra weight on deviations in ending recruitment stanza
  number f_fullF_constraint;        //penalty for Fapex>X
  number f_Ftune;                   //penalty for tuning F in Ftune yr.  Not applied in final optimization phase.
  number f_rec_RW_cpue;             //random walk component of index
  number f_priors;                  //prior information on parameters

  //init_number xdum;
  objective_function_value fval;
  number fval_data;
  number grad_max;

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

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


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

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

```
RUNTIME_SECTION
 maximum_function_evaluations 1000, 2000,3000, 10000;
 convergence_criteria 1e-2, 1e-2,1e-3, 1e-4;

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

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

  M=set_M;
  M_constant=set_M_constant(1);
  smsy2msst=0.75;  //more typically, 1-M_constant

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

  log_q_rec=set_log_q_rec(1);
  log_q_cH=set_log_q_cH(1);

  q_rate=set_q_rate;
  q_rate_fcn_rec=1.0;
  q_rate_fcn_cH=1.0;
  q_DD_beta=set_q_DD_beta;
  q_DD_fcn=1.0;

  q_RW_log_dev_rec.initialize();
  q_RW_log_dev_cH.initialize();

  if (set_q_rate_phase<0 & q_rate!=0.0)
  {
   for (iyear=styr_rec_cpue; iyear<=endyr_rec_cpue; iyear++)
     {   if (iyear>styr_rec_cpue & iyear <=2003)
         {//q_rate_fcn_rec(iyear)=(1.0+q_rate)*q_rate_fcn_rec(iyear-1); //compound
           q_rate_fcn_rec(iyear)=(1.0+(iyear-styr_rec_cpue)*q_rate)*q_rate_fcn_rec(styr_rec_cpue);  //linear
         }
         if (iyear>2003) {q_rate_fcn_rec(iyear)=q_rate_fcn_rec(iyear-1);}
     }
   for (iyear=styr_cH_cpue; iyear<=endyr_cH_cpue; iyear++)
     {   if (iyear>styr_cH_cpue & iyear <=2003)
         {//q_rate_fcn_cH(iyear)=(1.0+q_rate)*q_rate_fcn_cH(iyear-1); //compound
           q_rate_fcn_cH(iyear)=(1.0+(iyear-styr_cH_cpue)*q_rate)*q_rate_fcn_cH(styr_cH_cpue);  //linear
         }
         if (iyear>2003) {q_rate_fcn_cH(iyear)=q_rate_fcn_cH(iyear-1);}
     }

  } //end q_rate conditional

  w_L=set_w_L;

  w_I_cvt=set_w_I_cvt;
  w_I_vll=set_w_I_vll;
  w_I_rec=set_w_I_rec;
  w_I_cH=set_w_I_cH;

  w_lc_rec=set_w_lc_rec;
  w_lc_cH=set_w_lc_cH;
  w_lc_cL=set_w_lc_cL;

  w_ac_cvt=set_w_ac_cvt;
  w_ac_vll=set_w_ac_vll;
  w_ac_rec=set_w_ac_rec;
  w_ac_cH=set_w_ac_cH;
  w_ac_cL=set_w_ac_cL;

  w_Nage_init=set_w_Nage_init;
  w_rec=set_w_rec;
  w_rec_early=set_w_rec_early;
  w_rec_end=set_w_rec_end;
  w_fullF=set_w_fullF;
  w_Ftune=set_w_Ftune;

  F_init=set_F_init(1);
  log_avg_F_rec=set_log_avg_F_rec(1);
  log_avg_F_cH=set_log_avg_F_cH(1);
  log_avg_F_cL=set_log_avg_F_cL(1);

  log_F_dev_rec=set_log_F_dev_rec_vals;
  log_F_dev_cH=set_log_F_dev_cH_vals;
  log_F_dev_cL=set_log_F_dev_cL_vals;

  selpar_L50_cvt=set_selpar_L50_cvt(1);
  selpar_slope_cvt=set_selpar_slope_cvt(1);
  selpar_afull_cvt=set_selpar_afull_cvt(1);
  selpar_sigma_cvt=set_selpar_sigma_cvt(1);

  selpar_L50_vll=set_selpar_L50_vll(1);
```

```
 selpar_slope_vll=set_selpar_slope_vll(1);

 selpar_L50_cH=set_selpar_L50_cH(1);
 selpar_slope_cH=set_selpar_slope_cH(1);

 selpar_L50_cL=set_selpar_L50_cL(1);
 selpar_slope_cL=set_selpar_slope_cL(1);

 selpar_L50_rec=set_selpar_L50_rec(1);
 selpar_slope_rec=set_selpar_slope_rec(1);
 selpar_afull_rec=set_selpar_afull_rec(1);
 selpar_sigma_rec=set_selpar_sigma_rec(1);
 selpar_L50_rec2=set_selpar_L50_rec2(1);
 selpar_slope_rec2=set_selpar_slope_rec2(1);
 selpar_afull_rec2=set_selpar_afull_rec2(1);
 selpar_sigma_rec2=set_selpar_sigma_rec2(1);
 selpar_L50_rec3=set_selpar_L50_rec3(1);
 selpar_slope_rec3=set_selpar_slope_rec3(1);
 selpar_afull_rec3=set_selpar_afull_rec3(1);
 selpar_sigma_rec3=set_selpar_sigma_rec3(1);


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

 SSB_msy_out=0.0;

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

 maturity_f=maturity_f_obs;
 maturity_m=maturity_m_obs;
 prop_f=prop_f_obs;

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

      nsamp_rec_lenc_allyr=missing;
      nsamp_cH_lenc_allyr=missing;
      nsamp_cL_lenc_allyr=missing;
      nsamp_cvt_agec_allyr=missing;
      nsamp_vll_agec_allyr=missing;
      nsamp_rec_agec_allyr=missing;
      nsamp_cH_agec_allyr=missing;
      nsamp_cL_agec_allyr=missing;

      nsamp_rec_lenc_allyr=missing;
      nsamp_cH_lenc_allyr=missing;
      nsamp_cL_lenc_allyr=missing;
      nsamp_cvt_agec_allyr=missing;
      nsamp_vll_agec_allyr=missing;
      nsamp_rec_agec_allyr=missing;
      nsamp_cH_agec_allyr=missing;
      nsamp_cL_agec_allyr=missing;


      for (iyear=1; iyear<=nyr_rec_lenc; iyear++)
        {if (nsamp_rec_lenc(iyear)>=minSS_rec_lenc)
          {nsamp_rec_lenc_allyr(yrs_rec_lenc(iyear))=nsamp_rec_lenc(iyear);
           nfish_rec_lenc_allyr(yrs_rec_lenc(iyear))=nfish_rec_lenc(iyear);}}

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

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

      for (iyear=1; iyear<=nyr_cvt_agec; iyear++)
        {if (nsamp_cvt_agec(iyear)>=minSS_cvt_agec)
          {nsamp_cvt_agec_allyr(yrs_cvt_agec(iyear))=nsamp_cvt_agec(iyear);
           nfish_cvt_agec_allyr(yrs_cvt_agec(iyear))=nfish_cvt_agec(iyear);}}

      for (iyear=1; iyear<=nyr_vll_agec; iyear++)
        {if (nsamp_vll_agec(iyear)>=minSS_vll_agec)
          {nsamp_vll_agec_allyr(yrs_vll_agec(iyear))=nsamp_vll_agec(iyear);
           nfish_vll_agec_allyr(yrs_vll_agec(iyear))=nfish_vll_agec(iyear);}}

      for (iyear=1; iyear<=nyr_rec_agec; iyear++)
        {if (nsamp_rec_agec(iyear)>=minSS_rec_agec)
          {nsamp_rec_agec_allyr(yrs_rec_agec(iyear))=nsamp_rec_agec(iyear);
           nfish_rec_agec_allyr(yrs_rec_agec(iyear))=nfish_rec_agec(iyear);}}

      for (iyear=1; iyear<=nyr_cH_agec; iyear++)
```

```
        {if (nsamp_cH_agec(iyear)>=minSS_cH_agec)
          {nsamp_cH_agec_allyr(yrs_cH_agec(iyear))=nsamp_cH_agec(iyear);
          nfish_cH_agec_allyr(yrs_cH_agec(iyear))=nfish_cH_agec(iyear);}}

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

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


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

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

  sel_cvt.initialize();
  sel_vll.initialize();
  sel_rec.initialize();
  sel_cH.initialize();
  sel_cL.initialize();

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


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

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

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

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

  get_length_weight_at_age();
  //cout << "got length, weight, fecundity transitions" <<endl;
  get_reprod();
  //cout << "got repro stuff" << endl;
  get_length_at_age_dist();
  //cout<< "got predicted length at age distribution"<<endl;
  get_weight_at_age_landings();
  //cout<< "got weight at age of landings"<<endl;
  get_spr_F0();
  //cout << "got F0 spr" << endl;
  get_selectivity();
  //cout << "got selectivity" << endl;
  get_mortality();
  //cout << "got mortalities" << endl;
  get_bias_corr();
  //cout<< "got recruitment bias correction" << endl;
  get_numbers_at_age();
  //cout << "got numbers at age" << endl;
  get_landings_numbers();
  //cout << "got landings in numbers" << endl;
  get_landings_wgt();
  //cout << "got landings in wgt" << endl;
  get_catchability_fcns();
  //cout << "got catchability_fcns" << endl;
  get_indices();
  //cout << "got indices" << endl;
  get_length_comps();
  //cout<< "got length comps"<< endl;
  get_age_comps();
  //cout<< "got age comps"<< endl;
  evaluate_objective_function();
  //cout << "objective function calculations complete" << endl;


FUNCTION get_length_weight_at_age
```

```
  //compute mean length (mm TL) and weight (whole) at age
    meanlen_TL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));     //total length in mm
    wgt_kg=wgtpar_a*pow(meanlen_TL,wgtpar_b);             //wgt in kg
    wgt_g=wgt_kg/g2kg;                                    //convert wgt in kg to weight in g
    wgt_mt=wgt_g*g2mt;                                    //convert weight in g to weight in mt
    wgt_klb=mt2klb*wgt_mt;                                //1000 lb of whole wgt
    wgt_lb=mt2lb*wgt_mt;                                  //1000 lb of whole wgt

FUNCTION get_reprod
   //reprod is product of stuff going into reproductive capacity calcs
   reprod=elem_prod((elem_prod(prop_f,maturity_f)+elem_prod((1.0-prop_f),maturity_m)),wgt_mt);
   reprod2=elem_prod(elem_prod(prop_f,maturity_f),wgt_mt);

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

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

    zscore_lzero=(0.0-meanlen_TL(iage))/len_sd(iage);
    cprob_lzero=cumd_norm(zscore_lzero);

    //first length bin
    zscore_len=((lenbins(1)+0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
    cprob_lenvec(1)=cumd_norm(zscore_len);          //includes any probability mass below zero
    lenprob(iage,1)=cprob_lenvec(1)-cprob_lzero;    //removes any probability mass below zero

    //most other length bins
    for (ilen=2;ilen<nlenbins;ilen++)
      {
       zscore_len=((lenbins(ilen)+0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
       cprob_lenvec(ilen)=cumd_norm(zscore_len);
       lenprob(iage,ilen)=cprob_lenvec(ilen)-cprob_lenvec(ilen-1);
      }
    //last length bin is a plus group
    zscore_len=((lenbins(nlenbins)-0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
    lenprob(iage,nlenbins)=1.0-cumd_norm(zscore_len);

    lenprob(iage)=lenprob(iage)/(1.0-cprob_lzero);  //renormalize to account for any prob mass below size=0
  }
  //fleet and survey specific length probs, all assumed here to equal the popn
  lenprob_rec=lenprob;
  lenprob_cH=lenprob;
  lenprob_cL=lenprob;


FUNCTION get_weight_at_age_landings

  for (iyear=styr; iyear<=endyr; iyear++)
  {
    len_rec_mm(iyear)=meanlen_TL;
    wholewgt_rec_klb(iyear)=wgt_klb;
    len_cH_mm(iyear)=meanlen_TL;
    wholewgt_cH_klb(iyear)=wgt_klb;
    len_cL_mm(iyear)=meanlen_TL;
    wholewgt_cL_klb(iyear)=wgt_klb;
  }

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

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

FUNCTION get_selectivity

  for (iyear=styr; iyear<=endyr; iyear++)
  {
     sel_cvt(iyear)=logistic_exponential(agebins, selpar_L50_cvt, selpar_slope_cvt, selpar_sigma_cvt, selpar_afull_cvt);
     sel_vll(iyear)=logistic(agebins, selpar_L50_vll, selpar_slope_vll);
     sel_rec(iyear)=logistic_exponential(agebins, selpar_L50_rec2, selpar_slope_rec2, selpar_sigma_rec2, selpar_afull_rec2);
     sel_cH(iyear)=logistic(agebins, selpar_L50_cH, selpar_slope_cH);
     sel_cL(iyear)=logistic(agebins, selpar_L50_cL, selpar_slope_cL);

     sel_cvt(iyear)(15,nages)=sel_cvt(iyear)(14);
     sel_rec(iyear)(15,nages)=sel_rec(iyear)(14);
     sel_cH(iyear)(15,nages)=sel_cH(iyear)(14);
  }

  //BLOCK 1 for rec selex. In this case, code is faster to just replace the previously calculated vector
  for (iyear=styr; iyear<=endyr_selex_phase1; iyear++)
   {sel_rec(iyear)=logistic_exponential(agebins, selpar_L50_rec, selpar_slope_rec, selpar_sigma_rec, selpar_afull_rec);
```

```
   sel_rec(iyear)(15,nages)=sel_rec(iyear)(14);
   }

  //BLOCK 3 for rec selex. In this case, code is faster to just replace the previously calculated vector
  for (iyear=(endyr_selex_phase2+1); iyear<=endyr; iyear++)
   {sel_rec(iyear)=logistic_exponential(agebins, selpar_L50_rec3, selpar_slope_rec3, selpar_sigma_rec3, selpar_afull_rec3);
   sel_rec(iyear)(15,nages)=sel_rec(iyear)(14);
   }



FUNCTION get_mortality
  Fsum.initialize();
  Fapex.initialize();
  F.initialize();
  //initialization F is avg from first 3 yrs of observed landings: cL is only series that starts in 1962
  log_F_dev_init_rec=sum(log_F_dev_rec(styr_rec_L,(styr_rec_L+2)))/3.0;
  log_F_dev_init_cH=sum(log_F_dev_cH(styr_cH_L,(styr_cH_L+2)))/3.0;

  for (iyear=styr; iyear<=endyr; iyear++)
  {
   if(iyear>=styr_rec_L & iyear<=endyr_rec_L)
   {  F_rec_out(iyear)=mfexp(log_avg_F_rec+log_F_dev_rec(iyear)); //}
     F_rec(iyear)=sel_rec(iyear)*F_rec_out(iyear);
     Fsum(iyear)+=F_rec_out(iyear);
   }

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

   if(iyear>=styr_cL_L & iyear<=endyr_cL_L)
   {  F_cL_out(iyear)=mfexp(log_avg_F_cL+log_F_dev_cL(iyear)); //}
     F_cL(iyear)=sel_cL(iyear)*F_cL_out(iyear);
     Fsum(iyear)+=F_cL_out(iyear);
   }

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

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

  }  //end iyear


FUNCTION get_bias_corr
  var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_dev)-
            sum(log_rec_dev(styr_rec_dev,endyr_rec_dev))/nyrs_rec)
            /(nyrs_rec-1.0);
  rec_sigma_sq=square(rec_sigma);
  if (set_BiasCor <= 0.0) {BiasCor=mfexp(rec_sigma_sq/2.0);}   //bias correction based on Rsigma
  else {BiasCor=set_BiasCor;}

FUNCTION get_numbers_at_age
//Initialization
  R0=mfexp(log_R0);
  S0=spr_F0*R0;

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

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

  F_init_cH_prop=mfexp(log_avg_F_cH+log_F_dev_init_cH)/(mfexp(log_avg_F_cH+log_F_dev_init_cH)+mfexp(log_avg_F_rec+log_F_dev_init_rec));
  F_init_rec_prop=mfexp(log_avg_F_rec+log_F_dev_init_rec)/(mfexp(log_avg_F_cH+log_F_dev_init_cH)+mfexp(log_avg_F_rec+log_F_dev_init_rec));

  F_initial=sel_cH(styr)*F_init*F_init_cH_prop+
          sel_rec(styr)*F_init*F_init_rec_prop;  //comm cL not operating at styr
  Z_initial=M+F_initial;

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

  if (styr==styr_rec_dev) {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, 1.0, SR_switch);} //without bias correction (deviation added later)
  else {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, BiasCor, SR_switch);} //with bias correction
  if(R1<10.0) {R1=10.0;} //Avoid unrealistically low popn sizes during search algorithm

//Compute equilibrium age structure for first year
  N_initial_eq(1)=R1;
  for (iage=2; iage<=nages; iage++)
  {
```

```
    N_initial_eq(iage)=N_initial_eq(iage-1)*
       mfexp(-1.0*(Z_initial(iage-1)));
  }
  //plus group calculation
  N_initial_eq(nages)=N_initial_eq(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group

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

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

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

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

 //Rest of years
  for (iyear=styr; iyear<endyr; iyear++)
  {
    if(iyear<(styr_rec_dev-1)|iyear>(endyr_rec_dev-1)) //recruitment follows S-R curve (with bias correction) exactly
    {
       N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch);
       N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
       N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages)); //plus group
       N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
       N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));
       B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
    }
    else   //recruitment follows S-R curve with lognormal deviation
    {
       N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch)*mfexp(log_rec_dev(iyear+1));
       N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));
       N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages)); //plus group
       N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year
       N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time
       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));
       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));
       B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
    }
  }

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

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

 FUNCTION get_landings_numbers //Baranov catch eqn
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      L_rec_num(iyear,iage)=N(iyear,iage)*F_rec(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cH_num(iyear,iage)=N(iyear,iage)*F_cH(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
      L_cL_num(iyear,iage)=N(iyear,iage)*F_cL(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
    pred_rec_L_knum(iyear)=sum(L_rec_num(iyear))/1000.0;
    pred_cH_L_knum(iyear)=sum(L_cH_num(iyear))/1000.0;
    pred_cL_L_knum(iyear)=sum(L_cL_num(iyear))/1000.0;
  }


 FUNCTION get_landings_wgt
  for (iyear=styr; iyear<=endyr; iyear++)
  {
    L_rec_klb(iyear)=elem_prod(L_rec_num(iyear),wholewgt_rec_klb(iyear));  //in 1000 lb
    L_cH_klb(iyear)=elem_prod(L_cH_num(iyear),wholewgt_cH_klb(iyear));     //in 1000 lb
    L_cL_klb(iyear)=elem_prod(L_cL_num(iyear),wholewgt_cL_klb(iyear));     //in 1000 lb

    pred_rec_L_klb(iyear)=sum(L_rec_klb(iyear));
    pred_cH_L_klb(iyear)=sum(L_cH_klb(iyear));
    pred_cL_L_klb(iyear)=sum(L_cL_klb(iyear));

  }


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

      for (iyear=styr_rec_cpue; iyear<=endyr_rec_cpue; iyear++)
```

```
    {   if (iyear>styr_rec_cpue & iyear <=2003)
        {//q_rate_fcn_rec(iyear)=(1.0+q_rate)*q_rate_fcn_rec(iyear-1); //compound
          q_rate_fcn_rec(iyear)=(1.0+(iyear-styr_rec_cpue)*q_rate)*q_rate_fcn_rec(styr_rec_cpue);  //linear
        }
        if (iyear>2003) {q_rate_fcn_rec(iyear)=q_rate_fcn_rec(iyear-1);}
    }
    for (iyear=styr_cH_cpue; iyear<=endyr_cH_cpue; iyear++)
    {   if (iyear>styr_cH_cpue & iyear <=2003)
        {//q_rate_fcn_cH(iyear)=(1.0+q_rate)*q_rate_fcn_cH(iyear-1); //compound
          q_rate_fcn_cH(iyear)=(1.0+(iyear-styr_cH_cpue)*q_rate)*q_rate_fcn_cH(styr_cH_cpue);  //linear
        }
        if (iyear>2003) {q_rate_fcn_cH(iyear)=q_rate_fcn_cH(iyear-1);}
    }
 } //end q_rate conditional

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


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

 //Survey 1: Marmap cvt
  for (iyear=styr_cvt_cpue; iyear<=endyr_cvt_cpue; iyear++)
  {   //index in number units
     N_cvt(iyear)=elem_prod(N_mdyr(iyear),sel_cvt(iyear));
     pred_cvt_cpue(iyear)=mfexp(log_q_cvt)*sum(N_cvt(iyear));
  }

 //Survey 2: Marmap vll
  for (iyear=styr_vll_cpue; iyear<=endyr_vll_cpue; iyear++)
  {   //index in number units
     N_vll(iyear)=elem_prod(N_mdyr(iyear),sel_vll(iyear));
     pred_vll_cpue(iyear)=mfexp(log_q_vll)*sum(N_vll(iyear));
  }


 //rec  cpue
  q_rec(styr_rec_cpue)=mfexp(log_q_rec);
  for (iyear=styr_rec_cpue; iyear<=endyr_rec_cpue; iyear++)
  {
     N_rec(iyear)=elem_prod(N_mdyr(iyear),sel_rec(iyear));
     pred_rec_cpue(iyear)=q_rec(iyear)*q_rate_fcn_rec(iyear)*q_DD_fcn(iyear)*sum(N_rec(iyear));
     if (iyear<endyr_rec_cpue){q_rec(iyear+1)=q_rec(iyear)*mfexp(q_RW_log_dev_rec(iyear));}
  }

 //cH  cpue
  q_cH(styr_cH_cpue)=mfexp(log_q_cH);
  for (iyear=styr_cH_cpue; iyear<=endyr_cH_cpue; iyear++)
  {//index in weight units. original index in lb and re-scaled. predicted in klb, but difference is absorbed by q
     N_cH(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_cH(iyear)),wholewgt_cH_klb(iyear));
     pred_cH_cpue(iyear)=q_cH(iyear)*q_rate_fcn_cH(iyear)*q_DD_fcn(iyear)*sum(N_cH(iyear));
     if (iyear<endyr_cH_cpue){q_cH(iyear+1)=q_cH(iyear)*mfexp(q_RW_log_dev_cH(iyear));}
  }


FUNCTION get_length_comps

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

 //comm longline
  for (iyear=1;iyear<=nyr_cL_lenc;iyear++)
  {pred_cL_lenc(iyear)=(L_cL_num(yrs_cL_lenc(iyear))*lenprob_cL)/sum(L_cL_num(yrs_cL_lenc(iyear)));}

 //headboat
  for (iyear=1;iyear<=nyr_rec_lenc;iyear++)
  {pred_rec_lenc(iyear)=(L_rec_num(yrs_rec_lenc(iyear))*lenprob_rec)/sum(L_rec_num(yrs_rec_lenc(iyear)));}



FUNCTION get_age_comps

 //MM cvt
 for (iyear=1;iyear<=nyr_cvt_agec;iyear++)
  {
    ErrorFree_cvt_agec(iyear)=N_cvt(yrs_cvt_agec(iyear))/sum(N_cvt(yrs_cvt_agec(iyear)));
    pred_cvt_agec_allages(iyear)=age_error*ErrorFree_cvt_agec(iyear);
    for (iage=1; iage<=nages_agec; iage++) {pred_cvt_agec(iyear,iage)=pred_cvt_agec_allages(iyear,iage);}
    for (iage=(nages_agec+1); iage<=nages; iage++) {pred_cvt_agec(iyear,nages_agec)+=pred_cvt_agec_allages(iyear,iage);} //plus group
  }

 //MM vll
 for (iyear=1;iyear<=nyr_vll_agec;iyear++)
  {
    ErrorFree_vll_agec(iyear)=N_vll(yrs_vll_agec(iyear))/sum(N_vll(yrs_vll_agec(iyear)));
    pred_vll_agec_allages(iyear)=age_error*ErrorFree_vll_agec(iyear);
```

```
   for (iage=1; iage<=nages_agec; iage++) {pred_vll_agec(iyear,iage)=pred_vll_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++) {pred_vll_agec(iyear,nages_agec)+=pred_vll_agec_allages(iyear,iage);} //plus group
  }

 //Recreational
 for (iyear=1;iyear<=nyr_rec_agec;iyear++)
  {
   ErrorFree_rec_agec(iyear)=L_rec_num(yrs_rec_agec(iyear))/sum(L_rec_num(yrs_rec_agec(iyear)));
   pred_rec_agec_allages(iyear)=age_error*ErrorFree_rec_agec(iyear);
   for (iage=1; iage<=nages_agec; iage++) {pred_rec_agec(iyear,iage)=pred_rec_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++) {pred_rec_agec(iyear,nages_agec)+=pred_rec_agec_allages(iyear,iage);} //plus group
  }

  //Commercial handline
  for (iyear=1;iyear<=nyr_cH_agec;iyear++)
  {
   ErrorFree_cH_agec(iyear)=L_cH_num(yrs_cH_agec(iyear))/sum(L_cH_num(yrs_cH_agec(iyear)));
   pred_cH_agec_allages(iyear)=age_error*(ErrorFree_cH_agec(iyear)/sum(ErrorFree_cH_agec(iyear)));
   for (iage=1; iage<=nages_agec; iage++) {pred_cH_agec(iyear,iage)=pred_cH_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++) {pred_cH_agec(iyear,nages_agec)+=pred_cH_agec_allages(iyear,iage);} //plus group

  }

  //Commercial longline
  for (iyear=1;iyear<=nyr_cL_agec;iyear++)
  {
   ErrorFree_cL_agec(iyear)=L_cL_num(yrs_cL_agec(iyear))/sum(L_cL_num(yrs_cL_agec(iyear)));
   pred_cL_agec_allages(iyear)=age_error*(ErrorFree_cL_agec(iyear)/sum(ErrorFree_cL_agec(iyear)));
   for (iage=1; iage<=nages_agec; iage++) {pred_cL_agec(iyear,iage)=pred_cL_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++) {pred_cL_agec(iyear,nages_agec)+=pred_cL_agec_allages(iyear,iage);} //plus group

  }

////--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
FUNCTION get_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_rec+
       sum(log_F_dev_rec((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cH+
       sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cL+
       sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);

  F_rec_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_rec+
       sum(log_F_dev_rec((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cH_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cH+
       sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
  F_cL_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cL+
       sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;

  log_F_dev_end_rec=sum(log_F_dev_rec((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_cH=sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
  log_F_dev_end_cL=sum(log_F_dev_cL((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

  F_end_L=sel_rec(endyr)*mfexp(log_avg_F_rec+log_F_dev_end_rec)+
        sel_cH(endyr)*mfexp(log_avg_F_cH+log_F_dev_end_cH)+
        sel_cL(endyr)*mfexp(log_avg_F_cL+log_F_dev_end_cL);

  F_end=F_end_L;
  F_end_apex=max(F_end);

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

  wgt_wgted_L_denom=F_rec_prop+F_cH_prop+F_cL_prop;
  wgt_wgted_L_klb=F_rec_prop/wgt_wgted_L_denom*wholewgt_rec_klb(endyr)+
              F_cH_prop/wgt_wgted_L_denom*wholewgt_cH_klb(endyr)+
              F_cL_prop/wgt_wgted_L_denom*wholewgt_cL_klb(endyr);

FUNCTION get_msy

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

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

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

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

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

```
   if (R_eq(ff)<dzero) {R_eq(ff)=dzero;}
   N_age_msy*=R_eq(ff);
   N_age_msy_spawn*=R_eq(ff);

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

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

  msy_klb_out=max(L_eq_klb);

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

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

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

  len_cv=elem_div(len_sd,meanlen_TL);

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

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

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

  L_total_num=L_rec_num+L_cH_num+L_cL_num;   //landings at age in number fish
  L_total_klb=L_rec_klb+L_cH_klb+L_cH_klb;   //landings at age in klb whole weight

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

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

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


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

  //static per-recruit stuff

  for(iyear=styr; iyear<=endyr; iyear++)
  {
   N_age_spr(1)=1.0;
```

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


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

  F_L_age_spr=F_spr(ff)*sel_wgted_L;

  Z_age_spr=M+F_L_age_spr;

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

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

FUNCTION get_effective_sample_sizes
     neff_cH_lenc_allyr_out=missing;
     neff_cL_lenc_allyr_out=missing;
     neff_rec_lenc_allyr_out=missing;

     neff_cvt_agec_allyr_out=missing;
     neff_vll_agec_allyr_out=missing;
     neff_rec_agec_allyr_out=missing;
     neff_cH_agec_allyr_out=missing;
     neff_cL_agec_allyr_out=missing;

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

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

     for (iyear=1; iyear<=nyr_rec_lenc; iyear++)
       {if (nsamp_rec_lenc(iyear)>=minSS_rec_lenc)
         {neff_rec_lenc_allyr_out(yrs_rec_lenc(iyear))=multinom_eff_N(pred_rec_lenc(iyear),obs_rec_lenc(iyear));}
        else {neff_rec_lenc_allyr_out(yrs_rec_lenc(iyear))=-99;}
        }


     for (iyear=1; iyear<=nyr_cvt_agec; iyear++)
       {if (nsamp_cvt_agec(iyear)>=minSS_cvt_agec)
         {neff_cvt_agec_allyr_out(yrs_cvt_agec(iyear))=multinom_eff_N(pred_cvt_agec(iyear),obs_cvt_agec(iyear));}
        else {neff_cvt_agec_allyr_out(yrs_cvt_agec(iyear))=-99;}
        }

     for (iyear=1; iyear<=nyr_vll_agec; iyear++)
       {if (nsamp_vll_agec(iyear)>=minSS_vll_agec)
         {neff_vll_agec_allyr_out(yrs_vll_agec(iyear))=multinom_eff_N(pred_vll_agec(iyear),obs_vll_agec(iyear));}
        else {neff_vll_agec_allyr_out(yrs_vll_agec(iyear))=-99;}
        }

     for (iyear=1; iyear<=nyr_rec_agec; iyear++)
       {if (nsamp_rec_agec(iyear)>=minSS_rec_agec)
         {neff_rec_agec_allyr_out(yrs_rec_agec(iyear))=multinom_eff_N(pred_rec_agec(iyear),obs_rec_agec(iyear));}
        else {neff_rec_agec_allyr_out(yrs_rec_agec(iyear))=-99;}
        }

     for (iyear=1; iyear<=nyr_cH_agec; iyear++)
       {if (nsamp_cH_agec(iyear)>=minSS_cH_agec)
```

```
        {neff_cH_agec_allyr_out(yrs_cH_agec(iyear))=multinom_eff_N(pred_cH_agec(iyear),obs_cH_agec(iyear));}
      else {neff_cH_agec_allyr_out(yrs_cH_agec(iyear))=-99;}
      }

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

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

FUNCTION evaluate_objective_function
  //fval=square(xdum-9.0); //only used during code development

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

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

  f_cvt_cpue=0.0;
  f_cvt_cpue=lk_lognormal(pred_cvt_cpue, obs_cvt_cpue, cvt_cpue_cv, w_I_cvt);
  fval+=f_cvt_cpue;
  fval_data+=f_cvt_cpue;

  f_vll_cpue=0.0;
  f_vll_cpue=lk_lognormal(pred_vll_cpue, obs_vll_cpue, vll_cpue_cv, w_I_vll);
  fval+=f_vll_cpue;
  fval_data+=f_vll_cpue;

  f_rec_cpue=0.0;
  f_rec_cpue=lk_lognormal(pred_rec_cpue, obs_rec_cpue, rec_cpue_cv, w_I_rec);
  fval+=f_rec_cpue;
  fval_data+=f_rec_cpue;

  f_cH_cpue=0.0;
  fval+=f_cH_cpue;
  fval_data+=f_cH_cpue;

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

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

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

  //f_rec_L in 1000 fish
  f_rec_L=lk_lognormal(pred_rec_L_knum(styr_rec_L,endyr_rec_L), obs_rec_L(styr_rec_L,endyr_rec_L),
                rec_L_cv(styr_rec_L,endyr_rec_L), w_L);
  fval+=f_rec_L;
  fval_data+=f_rec_L;


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


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

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

  //f_rec_lenc
  f_rec_lenc=lk_robust_multinomial(nsamp_rec_lenc, pred_rec_lenc, obs_rec_lenc, nyr_rec_lenc, double(nlenbins), minSS_rec_lenc, w_lc_rec);
  //f_rec_lenc=lk_multinomial(nsamp_rec_lenc, pred_rec_lenc, obs_rec_lenc, nyr_rec_lenc, minSS_rec_lenc, w_lc_rec);
  fval+=f_rec_lenc;
  fval_data+=f_rec_lenc;


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

  //f_cvt_agec
  f_cvt_agec=lk_robust_multinomial(nsamp_cvt_agec, pred_cvt_agec, obs_cvt_agec, nyr_cvt_agec, double(nages_agec), minSS_cvt_agec, w_ac_cvt);
  //f_cvt_agec=lk_multinomial(nsamp_cvt_agec, pred_cvt_agec, obs_cvt_agec, nyr_cvt_agec, minSS_cvt_agec, w_ac_cvt);
  fval+=f_cvt_agec;
  fval_data+=f_cvt_agec;

  //f_vll_agec
```

```
  f_vll_agec=lk_robust_multinomial(nsamp_vll_agec, pred_vll_agec, obs_vll_agec, nyr_vll_agec, double(nages_agec), minSS_vll_agec, w_ac_vll);
  //f_vll_agec=lk_multinomial(nsamp_vll_agec, pred_vll_agec, obs_vll_agec, nyr_vll_agec, minSS_vll_agec, w_ac_vll);
  fval+=f_vll_agec;
  fval_data+=f_vll_agec;

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

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

  //f_rec_agec
  f_rec_agec=lk_robust_multinomial(nsamp_rec_agec, pred_rec_agec, obs_rec_agec, nyr_rec_agec, double(nages_agec), minSS_rec_agec, w_ac_rec);
  //f_rec_agec=lk_multinomial(nsamp_rec_agec, pred_rec_agec, obs_rec_agec, nyr_rec_agec, minSS_rec_agec, w_ac_rec);
  fval+=f_rec_agec;
  fval_data+=f_rec_agec;

//-----------Constraints and penalties--------------------------------

  //Light penalty applied to log_Nage_dev for deviation from zero. If not estimated, this penalty equals zero.
  f_Nage_init=norm2(log_Nage_dev);
  fval+=w_Nage_init*f_Nage_init;

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


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

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

  //Ftune penalty: does not apply in last phase
  f_Ftune=0.0;
  if (w_Ftune>0.0)
  {if (set_Ftune>0.0 && !last_phase()) {f_Ftune=square(Fapex(set_Ftune_yr)-set_Ftune);}
   fval+=w_Ftune*f_Ftune;
  }


  //Penalty if apical F exceeds 3.0
  f_fullF_constraint=0.0;
  if (w_fullF>0.0)
  {for (iyear=styr; iyear<=endyr; iyear++)
       {if(Fapex(iyear)>3.0) {f_fullF_constraint+=(mfexp(Fapex(iyear)-3.0)-1.0);}}
   fval+=w_fullF*f_fullF_constraint;
  }

  //Random walk components of fishery dependent indices
  f_rec_RW_cpue=0.0;
  for (iyear=styr_rec_cpue; iyear<endyr_rec_cpue; iyear++)
     {f_rec_RW_cpue+=square(q_RW_log_dev_rec(iyear))/(2.0*set_q_RW_rec_var);}
  fval+=f_rec_RW_cpue;


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

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

  f_priors+=neg_log_prior(selpar_L50_cvt,set_selpar_L50_cvt(5), set_selpar_L50_cvt(6), set_selpar_L50_cvt(7));
  f_priors+=neg_log_prior(selpar_slope_cvt,set_selpar_slope_cvt(5), set_selpar_slope_cvt(6), set_selpar_slope_cvt(7));
  f_priors+=neg_log_prior(selpar_afull_cvt,set_selpar_afull_cvt(5), set_selpar_afull_cvt(6), set_selpar_afull_cvt(7));
  f_priors+=neg_log_prior(selpar_sigma_cvt,set_selpar_sigma_cvt(5), set_selpar_sigma_cvt(6), set_selpar_sigma_cvt(7));

  f_priors+=neg_log_prior(selpar_L50_vll,set_selpar_L50_vll(5), set_selpar_L50_vll(6), set_selpar_L50_vll(7));
  f_priors+=neg_log_prior(selpar_slope_vll,set_selpar_slope_vll(5), set_selpar_slope_vll(6), set_selpar_slope_vll(7));

  f_priors+=neg_log_prior(selpar_L50_cH,set_selpar_L50_cH(5), set_selpar_L50_cH(6), set_selpar_L50_cH(7));
  f_priors+=neg_log_prior(selpar_slope_cH,set_selpar_slope_cH(5), set_selpar_slope_cH(6), set_selpar_slope_cH(7));

  f_priors+=neg_log_prior(selpar_L50_cL,set_selpar_L50_cL(5), set_selpar_L50_cL(6), set_selpar_L50_cL(7));
  f_priors+=neg_log_prior(selpar_slope_cL,set_selpar_slope_cL(5), set_selpar_slope_cL(6), set_selpar_slope_cL(7));

  f_priors+=neg_log_prior(selpar_L50_rec,set_selpar_L50_rec(5), set_selpar_L50_rec(6), set_selpar_L50_rec(7));
  f_priors+=neg_log_prior(selpar_slope_rec,set_selpar_slope_rec(5), set_selpar_slope_rec(6), set_selpar_slope_rec(7));
  f_priors+=neg_log_prior(selpar_afull_rec,set_selpar_afull_rec(5), set_selpar_afull_rec(6), set_selpar_afull_rec(7));
  f_priors+=neg_log_prior(selpar_sigma_rec,set_selpar_sigma_rec(5), set_selpar_sigma_rec(6), set_selpar_sigma_rec(7));

  f_priors+=neg_log_prior(selpar_L50_rec2,set_selpar_L50_rec2(5), set_selpar_L50_rec2(6), set_selpar_L50_rec2(7));
  f_priors+=neg_log_prior(selpar_slope_rec2,set_selpar_slope_rec2(5), set_selpar_slope_rec2(6), set_selpar_slope_rec2(7));
  f_priors+=neg_log_prior(selpar_afull_rec2,set_selpar_afull_rec2(5), set_selpar_afull_rec2(6), set_selpar_afull_rec2(7));
  f_priors+=neg_log_prior(selpar_sigma_rec2,set_selpar_sigma_rec2(5), set_selpar_sigma_rec2(6), set_selpar_sigma_rec2(7));

  f_priors+=neg_log_prior(selpar_L50_rec3,set_selpar_L50_rec3(5), set_selpar_L50_rec3(6), set_selpar_L50_rec3(7));
  f_priors+=neg_log_prior(selpar_slope_rec3,set_selpar_slope_rec3(5), set_selpar_slope_rec3(6), set_selpar_slope_rec3(7));
  f_priors+=neg_log_prior(selpar_afull_rec3,set_selpar_afull_rec3(5), set_selpar_afull_rec3(6), set_selpar_afull_rec3(7));
  f_priors+=neg_log_prior(selpar_sigma_rec3,set_selpar_sigma_rec3(5), set_selpar_sigma_rec3(6), set_selpar_sigma_rec3(7));

  f_priors+=neg_log_prior(F_init,set_F_init(5),set_F_init(6),set_F_init(7));

  fval+=f_priors;

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

//--------------------------------------------------------------------------------
//Logistic function: 2 parameters
FUNCTION dvar_vector logistic(const dvar_vector& ages, const dvariable& L50, const dvariable& slope)
  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1./(1.+mfexp(-1.*slope*(ages-L50))); //logistic;
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

//--------------------------------------------------------------------------------
//Logistic-exponential: 4 parameters (but 1 is fixed)
FUNCTION dvar_vector logistic_exponential(const dvar_vector& ages, const dvariable& L50, const dvariable& slope, const dvariable& sigma, const dvariable& joint)
  //ages=vector of ages, L50=age at 50% sel (ascending limb), slope=rate of increase, sigma=controls rate of descent (descending)
  //joint=age to join curves
  RETURN_ARRAYS_INCREMENT();
  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
  Sel_Tmp=1.0;
  for (iage=1; iage<=nages; iage++)
  {
   if (ages(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope*(ages(iage)-L50)));}
   if (ages(iage)>joint){Sel_Tmp(iage)=mfexp(-1.*square((ages(iage)-joint)/sigma));}
  }
  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
  RETURN_ARRAYS_DECREMENT();
  return Sel_Tmp;

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

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

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

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

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

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

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

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

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

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

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


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

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

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


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


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

```
        exit(0);
    }
    return LkvalTmp;


//--------------------------------------------------------------------------------
//SDNR: age comp likelihood (assumes fits are done with the robust multinomial function)
FUNCTION dvariable sdnr_multinomial(const double& ncomp, const dvar_vector& ages, const dvar_vector& nsamp,
                                const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const dvariable& wgt_dat)
  //ncomp=number of years of data, ages=vector of ages, nsamp=vector of N's,
  //pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, wgt_dat=likelihood weight for data source
  RETURN_ARRAYS_INCREMENT();
  dvariable SdnrTmp;
  dvar_vector o(1,ncomp);
  dvar_vector p(1,ncomp);
  dvar_vector ose(1,ncomp);
  dvar_vector res(1,ncomp);
  SdnrTmp=0.0;
  for (int ii=1; ii<=ncomp; ii++)
  {
    o(ii)=sum(elem_prod(ages,obs_comp(ii)));
    p(ii)=sum(elem_prod(ages,pred_comp(ii)));
    ose(ii)=sqrt((sum(elem_prod(square(ages),pred_comp(ii)))-square(p(ii)))/(nsamp(ii)*wgt_dat));
  }
  res=elem_div((o-p),ose);
  SdnrTmp=sqrt(sum(square(res-(sum(res)/ncomp))/(ncomp-1.0)));
  RETURN_ARRAYS_DECREMENT();
  return SdnrTmp;



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

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

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


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

     cout <<endl;
     cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;
//     cout << "BC Fmsy=" << F_msy_out<< "   BC SSBmsy=" << SSB_msy_out <<endl;
//     cout <<"F status="<<FdF_msy_end<<endl;
//     cout <<"Pop status="<<SdSSB_msy_end<<endl;
     cout << "h="<<steep<<"   R0="<<R0<<endl;
     //cout << "len_cv = "<<len_cv_val<<endl;
     cout << "><>--><>--><>--><>--><>--><>--><>--><>--><>"  <<endl;
     report << "TotalLikelihood " << fval << endl;
     report << "N" << endl;
     report << N<<endl;
     report << "F" << endl;
     report << F <<endl;

     sdnr_lc_cH=sdnr_multinomial(nyr_cH_lenc, lenbins, nsamp_cH_lenc, pred_cH_lenc, obs_cH_lenc, w_lc_cH);
```

```
  sdnr_lc_cL=sdnr_multinomial(nyr_cL_lenc, lenbins, nsamp_cL_lenc, pred_cL_lenc, obs_cL_lenc, w_lc_cL);
  sdnr_lc_rec=sdnr_multinomial(nyr_rec_lenc, lenbins, nsamp_rec_lenc, pred_rec_lenc, obs_rec_lenc, w_lc_rec);

  sdnr_ac_cvt=sdnr_multinomial(nyr_cvt_agec, agebins_agec, nsamp_cvt_agec, pred_cvt_agec, obs_cvt_agec, w_ac_cvt);
  sdnr_ac_vll=sdnr_multinomial(nyr_vll_agec, agebins_agec, nsamp_vll_agec, pred_vll_agec, obs_vll_agec, w_ac_vll);
  sdnr_ac_rec=sdnr_multinomial(nyr_rec_agec, agebins_agec, nsamp_rec_agec, pred_rec_agec, obs_rec_agec, w_ac_rec);
  sdnr_ac_cH=sdnr_multinomial(nyr_cH_agec, agebins_agec, nsamp_cH_agec, pred_cH_agec, obs_cH_agec, w_ac_cH);
  sdnr_ac_cL=sdnr_multinomial(nyr_cL_agec, agebins_agec, nsamp_cL_agec, pred_cL_agec, obs_cL_agec, w_ac_cL);

  sdnr_I_cvt=sdnr_lognormal(pred_cvt_cpue, obs_cvt_cpue, cvt_cpue_cv, w_I_cvt);
  sdnr_I_vll=sdnr_lognormal(pred_vll_cpue, obs_vll_cpue, vll_cpue_cv, w_I_vll);
  sdnr_I_rec=sdnr_lognormal(pred_rec_cpue, obs_rec_cpue, rec_cpue_cv, w_I_rec);
  sdnr_I_cH=1.0; //sdnr_lognormal(pred_cH_cpue, obs_cH_cpue, cH_cpue_cv, w_I_cH);


  //#########################################################################################
  //##  Passing parameters to vector for bounds check plotting
  //#########################################################################################
   Linf_out(8)=Linf; Linf_out(1,7)=set_Linf;
   K_out(8)=K; K_out(1,7)=set_K;
   t0_out(8)=t0; t0_out(1,7)=set_t0;
   len_cv_val_out(8)=len_cv_val; len_cv_val_out(1,7)=set_len_cv;
   log_R0_out(8)=log_R0; log_R0_out(1,7)=set_log_R0;
   M_constant_out(8)=M_constant; M_constant_out(1,7)=set_M_constant;
   steep_out(8)=steep; steep_out(1,7)=set_steep;
   rec_sigma_out(8)=rec_sigma; rec_sigma_out(1,7)=set_rec_sigma;
   R_autocorr_out(8)=R_autocorr; R_autocorr_out(1,7)=set_R_autocorr;

   selpar_L50_cvt_out(8)=selpar_L50_cvt; selpar_L50_cvt_out(1,7)=set_selpar_L50_cvt;
   selpar_slope_cvt_out(8)=selpar_slope_cvt; selpar_slope_cvt_out(1,7)=set_selpar_slope_cvt;
   selpar_afull_cvt_out(8)=selpar_afull_cvt; selpar_afull_cvt_out(1,7)=set_selpar_afull_cvt;
   selpar_sigma_cvt_out(8)=selpar_sigma_cvt; selpar_sigma_cvt_out(1,7)=set_selpar_sigma_cvt;

   selpar_L50_vll_out(8)=selpar_L50_vll; selpar_L50_vll_out(1,7)=set_selpar_L50_vll;
   selpar_slope_vll_out(8)=selpar_slope_vll; selpar_slope_vll_out(1,7)=set_selpar_slope_vll;

   selpar_L50_cH_out(8)=selpar_L50_cH; selpar_L50_cH_out(1,7)=set_selpar_L50_cH;
   selpar_slope_cH_out(8)=selpar_slope_cH; selpar_slope_cH_out(1,7)=set_selpar_slope_cH;

   selpar_L50_cL_out(8)=selpar_L50_cL; selpar_L50_cL_out(1,7)=set_selpar_L50_cL;
   selpar_slope_cL_out(8)=selpar_slope_cL; selpar_slope_cL_out(1,7)=set_selpar_slope_cL;

   selpar_L50_rec_out(8)=selpar_L50_rec; selpar_L50_rec_out(1,7)=set_selpar_L50_rec;
   selpar_slope_rec_out(8)=selpar_slope_rec; selpar_slope_rec_out(1,7)=set_selpar_slope_rec;
   selpar_afull_rec_out(8)=selpar_afull_rec; selpar_afull_rec_out(1,7)=set_selpar_afull_rec;
   selpar_sigma_rec_out(8)=selpar_sigma_rec; selpar_sigma_rec_out(1,7)=set_selpar_sigma_rec;
   selpar_L50_rec2_out(8)=selpar_L50_rec2; selpar_L50_rec2_out(1,7)=set_selpar_L50_rec2;
   selpar_slope_rec2_out(8)=selpar_slope_rec2; selpar_slope_rec2_out(1,7)=set_selpar_slope_rec2;
   selpar_afull_rec2_out(8)=selpar_afull_rec2; selpar_afull_rec2_out(1,7)=set_selpar_afull_rec2;
   selpar_sigma_rec2_out(8)=selpar_sigma_rec2; selpar_sigma_rec2_out(1,7)=set_selpar_sigma_rec2;
   selpar_L50_rec3_out(8)=selpar_L50_rec3; selpar_L50_rec3_out(1,7)=set_selpar_L50_rec3;
   selpar_slope_rec3_out(8)=selpar_slope_rec3; selpar_slope_rec3_out(1,7)=set_selpar_slope_rec3;
   selpar_afull_rec3_out(8)=selpar_afull_rec3; selpar_afull_rec3_out(1,7)=set_selpar_afull_rec3;
   selpar_sigma_rec3_out(8)=selpar_sigma_rec3; selpar_sigma_rec3_out(1,7)=set_selpar_sigma_rec3;

   log_q_cvt_out(8)=log_q_cvt; log_q_cvt_out(1,7)=set_log_q_cvt;
   log_q_vll_out(8)=log_q_vll; log_q_vll_out(1,7)=set_log_q_vll;
   log_q_rec_out(8)=log_q_rec; log_q_rec_out(1,7)=set_log_q_rec;
   log_q_cH_out(8)=log_q_cH; log_q_cH_out(1,7)=set_log_q_cH;

   log_avg_F_rec_out(8)=log_avg_F_rec; log_avg_F_rec_out(1,7)=set_log_avg_F_rec;
   log_avg_F_cH_out(8)=log_avg_F_cH; log_avg_F_cH_out(1,7)=set_log_avg_F_cH;
   log_avg_F_cL_out(8)=log_avg_F_cL; log_avg_F_cL_out(1,7)=set_log_avg_F_cL;
   F_init_out(8)=F_init; F_init_out(1,7)=set_F_init;

   log_rec_dev_out(styr_rec_dev, endyr_rec_dev)=log_rec_dev;
   log_F_dev_rec_out(styr_rec_L,endyr_rec_L)=log_F_dev_rec;
   log_F_dev_cH_out(styr_cH_L,endyr_cH_L)=log_F_dev_cH;
   log_F_dev_cL_out(styr_cL_L,endyr_cL_L)=log_F_dev_cL;

  #include "sg_make_Robject12.cxx"   // write the R-compatible report

} //endl last phase loop
```

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  SEDAR 36 SA snowy grouper assessment July 2013
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

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

#Starting and ending year of model
1974
2012

#Starting year to estimate recruitment deviation from S-R curve
1974

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

#3 phases of constraints on recruitment deviations:
#allows possible heavier constraint (weights defined later) in early and late period, with lighter constraint in the middle
#ending years of recruitment constraint phases
1992
2009

#Ending year for selectivity blocks, applied to rec selex only
1977
1991

#Number of ages in population model(25 classes are 1,...,N+) //assumes last age is plus group
25
#Vector of agebins, last is a plus group
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0
14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0
#Number of ages used to match age comps: first age must be same as popn, plus group may differ
14
#Vector of agebins, last is a plus group
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0

#Number length bins used to match length comps and width of bins
30   #number bins
30.0 #width of bins (mm)
#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group
220.0 250.0 280.0 310.0 340.0 370.0 400.0 430.0 460.0 490.0 520.0 550.0 580.0
610.0 640.0 670.0 700.0 730.0 760.0 790.0 820.0 850.0 880.0 910.0 940.0
970.0 1000.0 1030.0 1060.0 1090.0

#Max value of F used in spr and msy calculations
1.0
#Number of iterations in spr calculations
10001
#Number of iterations in msy calculations
10001
#Number years at end of time series over which to average sector Fs, for weighted selectivities
3
#Multiplicative bias correction of recruitment (may set to 1.0 for none or negative to compute from recruitment variance)
-1.0

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

######################## MM CVT index ##########################################################################
#Starting and ending years of index
1996
2012
#Observed CPUE and CVs
0.70 1.04 0.59 1.36 0.14 2.04 3.50 1.10 0.69 1.36 0.77 0.89 0.15
0.43 0.75 0.46 1.03
0.42 0.33 0.39 0.50 0.81 0.21 0.29 0.27 0.52 0.65 0.29 0.55 0.92
0.54 0.35 0.38 0.28
#Number and vector of years of age compositions for MMcvt
5
1996 1997 2001 2002 2012
#sample size of MMCVT age comp data by year  (first row observed Ntrips, second row Nfish)
20.0 18.0 13.0 10.0 23.0
56.0 61.0 38.0 28.0 44.0
#MMcvt age composition (year, age)
0.0000 0.0357 0.2500 0.3393 0.2321 0.1071 0.0357 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000
0.0492 0.1967 0.2459 0.3443 0.0492 0.0984 0.0000 0.0164 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.1579 0.1579 0.1842 0.3684 0.1316 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0714 0.4643 0.2143 0.0714 0.1429 0.0000 0.0000 0.0357 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0909 0.1364 0.4318 0.2045 0.0682 0.0682 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000

######################## MM vertical longline index ############################################################
#Starting and ending years of index
```

1996
2011
#Observed CPUE and CVs
0.47 0.70 0.68 0.93 0.70 1.00 1.38 0.97 0.42 1.01 0.67 1.35 1.78
2.24 0.93 0.78
0.42 0.28 0.38 0.27 0.26 0.25 0.29 0.20 0.70 0.23 0.26 0.28 0.18
0.26 0.19 0.33

#Number and vector of years of age compositions for MM vll
11
1997 1999 2000 2001 2002 2003 2005 2006 2008 2010 2011
#sample size of MM vll age comp data by year (first row observed Ntrips, second row Nfish)
14.0 14.0 19.0 18.0 10.0 25.0 19.0 15.0 20.0 43.0 57.0
38.0 33.0 36.0 42.0 27.0 52.0 36.0 30.0 61.0 98.0 127.0

#MM vll age composition samples (year, ages)
0.0000 0.0000 0.0789 0.1316 0.2105 0.0789 0.1316 0.1316 0.0526 0.0789 0.0000 0.0000 0.0789
0.0263
0.0000 0.0000 0.0000 0.0303 0.3636 0.3939 0.1515 0.0000 0.0303 0.0000 0.0000 0.0303 0.0000
0.0000
0.0000 0.0000 0.1667 0.1111 0.2500 0.1667 0.1111 0.0833 0.0556 0.0278 0.0000 0.0000 0.0278
0.0000
0.0000 0.0000 0.0000 0.0952 0.2381 0.2381 0.2143 0.0714 0.0714 0.0476 0.0238 0.0000 0.0000
0.0000
0.0000 0.0370 0.2963 0.1111 0.0370 0.1481 0.2222 0.0741 0.0741 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0000 0.1154 0.1731 0.2692 0.2308 0.0192 0.0385 0.0769 0.0192 0.0385 0.0000 0.0192
0.0000
0.0000 0.0556 0.1111 0.2778 0.1111 0.3056 0.0556 0.0556 0.0000 0.0278 0.0000 0.0000 0.0000
0.0000
0.0000 0.0000 0.1000 0.1333 0.4333 0.1667 0.0667 0.0667 0.0333 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0000 0.0000 0.0820 0.2131 0.2787 0.1967 0.0984 0.0492 0.0656 0.0000 0.0000 0.0164
0.0000
0.0000 0.0204 0.0510 0.1224 0.1531 0.1122 0.1837 0.1020 0.1429 0.0408 0.0204 0.0204 0.0000
0.0306
0.0000 0.0079 0.0709 0.0787 0.0945 0.1024 0.1102 0.1575 0.1339 0.1575 0.0315 0.0472 0.0000
0.0079

#######################Recreational ##
#Recreational headboat Index
#Starting and ending years of CPUE index
1978
2010
#Observed CPUE and CVs
1.58 1.22 2.38 2.18 0.97 1.26 0.85 0.84 0.87 1.17 1.11 1.39 0.93
1.02 0.68 0.49 0.57 0.77 0.96 0.75 0.72 0.80 0.75 0.92 1.08
1.36 0.54 0.64 0.96 0.91 0.54 0.94 0.85
0.14 0.15 0.13 0.15 0.11 0.09 0.12 0.10 0.10 0.11 0.12 0.10 0.15
0.14 0.14 0.12 0.11 0.16 0.14 0.23 0.17 0.21 0.17 0.17 0.34
0.35 0.13 0.17 0.31 0.22 0.18 0.16 0.25

#Total recreational landings + dead discards (HB + MRIP)
#Starting and ending years of landings time series, respectively
1974
2012
#Observed Total Recreational removals (1000s of fish) and assumed CVs (HB + MRIP + dead discards)
5.30 3.07 7.34 3.42 2.35 3.37 6.69 9.00 5.55 10.03 2.37 3.27 7.90
5.31 3.73 3.30 1.77 0.78 3.36 9.66 0.96 10.73 2.01 20.32 0.54
8.72 3.44 16.65 5.81 5.85 13.71 22.51 18.86 5.80 3.23 7.26 6.06
0.25 19.39
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05

#Number and vector of years of length compositions for recreational fishery (HB+MRIP)
18
1974 1975 1976 1977 1978 1979 1980 1983 1984 1985 1986 1987 1988
1989 1996 1997 1998 2001
#sample size of recreational length comp data by year (first row observed Ntrips, second row Nfish)
45.0 37.0 49.0 16.0 18.0 13.0 16.0 27.0 16.0 36.0 29.0 19.0 19.0
17.0 21.0 26.0 17.0 8.0
242.0 196.0 233.0 122.0 51.0 48.0 54.0 75.0 43.0 72.0 77.0 36.0 47.0
51.0 108.0 144.0 69.0 49.0
#recreational length composition samples (year,lengthbin 3 cm)
0.0000 0.0000 0.0000 0.0124 0.0124 0.0207 0.0289 0.0413 0.0455 0.0496 0.0455 0.0868 0.0702
0.0537 0.0455 0.0455 0.0620 0.0661 0.0579 0.0702 0.0620 0.0496 0.0289 0.0165 0.0124
0.0041 0.0083 0.0000 0.0000 0.0041
0.0000 0.0000 0.0255 0.0306 0.0153 0.0357 0.0153 0.0204 0.0408 0.0357 0.0714 0.0816 0.0255
0.0408 0.0408 0.0663 0.0714 0.0663 0.0714 0.0459 0.0357 0.0357 0.0612 0.0204 0.0102
0.0051 0.0204 0.0051 0.0000 0.0051
0.0043 0.0000 0.0000 0.0086 0.0343 0.0129 0.0773 0.0944 0.0429 0.0515 0.0858 0.0644 0.0558
0.0644 0.0815 0.0730 0.0515 0.0343 0.0258 0.0258 0.0172 0.0215 0.0215 0.0215 0.0129
0.0043 0.0000 0.0086 0.0043 0.0000
0.0000 0.0082 0.0000 0.0246 0.0164 0.0246 0.0738 0.0738 0.1311 0.0820 0.0738 0.0902 0.0820
0.0984 0.0574 0.0492 0.0246 0.0164 0.0246 0.0082 0.0082 0.0082 0.0000 0.0082 0.0082
0.0082 0.0000 0.0000 0.0000 0.0000
0.0392 0.0000 0.0588 0.0392 0.0196 0.0196 0.0784 0.0784 0.0392 0.0980 0.1569 0.1765 0.0588
0.0588 0.0196 0.0392 0.0196 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0208 0.0833 0.0417 0.0000 0.0833 0.1042 0.0208 0.1458 0.1250
0.1250 0.0833 0.0208 0.0208 0.0417 0.0208 0.0000 0.0208 0.0417 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0185 0.0000 0.0000 0.0000 0.0185 0.0556 0.0741 0.1481 0.0741 0.0926 0.0556 0.1296
0.0370 0.0370 0.0370 0.0741 0.0370 0.0185 0.0370 0.0000 0.0000 0.0185 0.0000 0.0185
0.0185 0.0000 0.0000 0.0000 0.0000
0.0133 0.0133 0.0133 0.0400 0.1067 0.0267 0.0533 0.0933 0.1600 0.1467 0.0133 0.0667 0.0533
0.0400 0.0400 0.0267 0.0133 0.0133 0.0000 0.0000 0.0133 0.0267 0.0000 0.0133 0.0133
0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.1628 0.0698 0.1395 0.0465 0.0698 0.1163 0.0698 0.0698 0.0465 0.1395
0.0000 0.0000 0.0233 0.0000 0.0000 0.0233 0.0233 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0694 0.0833 0.0556 0.1528 0.1111 0.0833 0.0556 0.1250 0.0417 0.0139 0.0417 0.0417
0.0278 0.0139 0.0139 0.0000 0.0278 0.0278 0.0000 0.0139 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0130 0.0649 0.0649 0.0649 0.1558 0.1039 0.1039 0.1299 0.0649 0.0649 0.0519 0.0519 0.0130
0.0130 0.0260 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0130 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0278 0.0278 0.1111 0.1111 0.0833 0.0556 0.0278 0.1667 0.0833 0.1389 0.0833 0.0278
0.0556 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0213 0.0213 0.0638 0.0638 0.0851 0.0638 0.1277 0.0638 0.0638 0.0426 0.2128 0.0851 0.0213
0.0000 0.0213 0.0213 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0213 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0784 0.1373 0.0980 0.1176 0.0000 0.0588 0.0196 0.0588 0.0980 0.0784
0.0980 0.0196 0.0588 0.0392 0.0196 0.0000 0.0196 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0185 0.0000 0.0278 0.0463 0.0185 0.0648 0.0926 0.1296 0.1481 0.0556 0.0833 0.0556 0.0185
0.0463 0.0185 0.0370 0.0463 0.0463 0.0185 0.0000 0.0093 0.0000 0.0093 0.0000 0.0093
0.0000 0.0000 0.0000 0.0000 0.0000
0.0069 0.0139 0.0000 0.0347 0.0278 0.0764 0.0486 0.0833 0.1042 0.1181 0.1389 0.0972 0.0694
0.0556 0.0208 0.0278 0.0208 0.0208 0.0139 0.0139 0.0000 0.0069 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000
0.0145 0.0000 0.0000 0.0145 0.0580 0.0870 0.0145 0.0435 0.0725 0.1449 0.0580 0.0870 0.1159
0.0435 0.0435 0.0145 0.0000 0.0580 0.0145 0.0435 0.0000 0.0000 0.0000 0.0000 0.0435
0.0145 0.0000 0.0000 0.0145 0.0000
0.0000 0.0204 0.0408 0.0000 0.1020 0.1224 0.0000 0.0408 0.1224 0.1429 0.1429 0.0408 0.0612
0.0612 0.0408 0.0204 0.0204 0.0204 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000

#Number and vector of years of age compositions for recreational fleet (headboat + mrip)
3
1981 2003 2004
#sample sizes of age comps by year (first row observed Ntrips, second row Nfish)
9.0 13.0 16.0
45.0 185.0 62.0
#age composition samples (year,age) from headboat fleet
0.0000 0.0444 0.0000 0.1111 0.0889 0.3111 0.1778 0.1111 0.0222 0.0889 0.0444 0.0000 0.0000
0.0000
0.0000 0.0324 0.2162 0.4432 0.2054 0.0649 0.0162 0.0108 0.0000 0.0054 0.0000 0.0000 0.0000
0.0054
0.0000 0.0484 0.1613 0.3871 0.0806 0.2258 0.0806 0.0161 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000

##Commercial Handline ##
#Comm handline Index
#Starting and ending years of CPUE index
1993
2005
#Observed CPUE and CVs
0.77 0.66 0.70 0.82 0.96 1.04 1.40 1.05 0.94 0.93 1.16 1.34 1.24
0.07 0.06 0.05 0.05 0.05 0.05 0.06 0.06 0.05 0.05 0.05 0.06 0.07

##Starting and ending years for landings time series
1974
2012
#commercial handline landings vector (1000 lb whole weight) and assumed CVs (landings + discards + kept for bait)
187.17 216.42 278.83 258.19 422.47 383.35 313.31 575.65 425.88 511.62 359.69 305.28 316.44
240.63 180.22 334.53 384.72 336.50 355.71 253.02 179.22 260.72 235.52 340.52 226.25
335.82 263.02 247.09 223.83 184.44 177.86 187.87 186.31 111.79 66.99 71.95 88.20
39.84 93.39
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05

#Number and vector of years of commercial hl length compositions
17
1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995
1996 1999 2000 2001
#Sample size of length comp data (first row observed Ntrips, second row Nfish)
8.0 84.0 136.0 110.0 90.0 82.0 91.0 83.0 102.0 111.0 108.0 92.0 137.0
105.0 115.0 132.0 124.0
95.0 2098.0 3645.0 1625.0 1395.0 795.0 1279.0 1677.0 1659.0 2997.0 2339.0 1922.0 4544.0
2143.0 2401.0 2261.0 1785.0
#commercial handline length comps (3 cm length bins)
0.0000 0.0000 0.0000 0.0211 0.0526 0.0421 0.0842 0.0947 0.1684 0.1368 0.0526 0.0316 0.0632
0.0737 0.0526 0.0316 0.0421 0.0105 0.0105 0.0000 0.0105 0.0000 0.0000 0.0000 0.0000
0.0000 0.0211 0.0000 0.0000 0.0000
0.0000 0.0008 0.0028 0.0090 0.0129 0.0222 0.0353 0.0465 0.0703 0.0638 0.0811 0.0681 0.0737
0.0542 0.0615 0.0411 0.0285 0.0204 0.0355 0.0385 0.0376 0.0414 0.0421 0.0492 0.0194
0.0175 0.0149 0.0044 0.0031 0.0042

0.0001 0.0029 0.0045 0.0162 0.0395 0.0360 0.0461 0.0518 0.0858 0.0861 0.0765 0.0848 0.0796
0.0559 0.0449 0.0283 0.0254 0.0111 0.0183 0.0247 0.0171 0.0183 0.0226 0.0259 0.0238
0.0308 0.0210 0.0061 0.0118 0.0041
0.0024 0.0045 0.0073 0.0215 0.0414 0.0635 0.0867 0.0976 0.0771 0.0825 0.0684 0.0363 0.0441
0.0271 0.0457 0.0330 0.0347 0.0230 0.0399 0.0362 0.0321 0.0189 0.0166 0.0043 0.0106
0.0140 0.0138 0.0065 0.0039 0.0063
0.0000 0.0015 0.0112 0.0400 0.0654 0.0919 0.1043 0.1120 0.0997 0.0958 0.1141 0.0751 0.0506
0.0342 0.0285 0.0177 0.0035 0.0065 0.0081 0.0056 0.0067 0.0148 0.0045 0.0008 0.0029
0.0030 0.0012 0.0004 0.0000 0.0000
0.0004 0.0022 0.0118 0.0285 0.0389 0.0813 0.1094 0.0968 0.1107 0.0603 0.0699 0.0851 0.0715
0.0491 0.0224 0.0278 0.0098 0.0186 0.0122 0.0150 0.0150 0.0064 0.0030 0.0150 0.0150
0.0090 0.0060 0.0030 0.0030 0.0030
0.0003 0.0031 0.0222 0.0638 0.0799 0.1097 0.1305 0.1622 0.0920 0.0739 0.0720 0.0512 0.0395
0.0250 0.0213 0.0102 0.0028 0.0031 0.0051 0.0020 0.0060 0.0020 0.0033 0.0050 0.0033
0.0037 0.0020 0.0033 0.0000 0.0017
0.0004 0.0021 0.0097 0.0161 0.0363 0.0547 0.1023 0.1292 0.1071 0.1151 0.0903 0.0716 0.0554
0.0651 0.0317 0.0303 0.0187 0.0156 0.0140 0.0061 0.0032 0.0059 0.0012 0.0079 0.0024
0.0028 0.0020 0.0020 0.0004 0.0004
0.0016 0.0049 0.0201 0.0472 0.0639 0.0699 0.0820 0.0753 0.0987 0.0842 0.0907 0.0694 0.0540
0.0371 0.0333 0.0291 0.0219 0.0222 0.0179 0.0119 0.0122 0.0040 0.0115 0.0069 0.0124
0.0082 0.0055 0.0027 0.0007 0.0007
0.0013 0.0000 0.0093 0.0255 0.0412 0.0620 0.0889 0.1006 0.0951 0.0952 0.0897 0.0771 0.0629
0.0464 0.0384 0.0253 0.0185 0.0185 0.0195 0.0101 0.0091 0.0094 0.0100 0.0141 0.0140
0.0084 0.0061 0.0015 0.0007 0.0013
0.0000 0.0032 0.0136 0.0223 0.0338 0.0412 0.0737 0.0909 0.0996 0.1008 0.0921 0.0934 0.0799
0.0606 0.0476 0.0262 0.0224 0.0156 0.0157 0.0074 0.0127 0.0073 0.0131 0.0118 0.0082
0.0034 0.0015 0.0008 0.0011 0.0000
0.0004 0.0051 0.0155 0.0416 0.0678 0.0956 0.0832 0.0967 0.1180 0.1079 0.0925 0.0744 0.0538
0.0436 0.0251 0.0266 0.0106 0.0098 0.0076 0.0065 0.0022 0.0029 0.0029 0.0022 0.0015
0.0026 0.0015 0.0011 0.0007 0.0004
0.0006 0.0019 0.0076 0.0214 0.0338 0.0633 0.0848 0.1066 0.1294 0.1086 0.0966 0.0789 0.0729
0.0482 0.0414 0.0280 0.0219 0.0130 0.0118 0.0081 0.0062 0.0047 0.0027 0.0013 0.0032
0.0016 0.0008 0.0002 0.0004 0.0002
0.0006 0.0024 0.0103 0.0380 0.0633 0.0799 0.0979 0.1345 0.1271 0.1182 0.0968 0.0735 0.0396
0.0272 0.0186 0.0166 0.0134 0.0113 0.0074 0.0081 0.0039 0.0026 0.0037 0.0000 0.0015
0.0009 0.0011 0.0009 0.0003 0.0006
0.0000 0.0015 0.0056 0.0177 0.0325 0.0417 0.0690 0.0978 0.0926 0.1024 0.0800 0.0682 0.0623
0.0541 0.0450 0.0323 0.0320 0.0151 0.0134 0.0174 0.0155 0.0181 0.0170 0.0160 0.0127
0.0158 0.0110 0.0093 0.0026 0.0014
0.0004 0.0019 0.0042 0.0187 0.0416 0.0696 0.0742 0.1019 0.1021 0.1064 0.1020 0.0747 0.0448
0.0443 0.0337 0.0225 0.0164 0.0139 0.0098 0.0038 0.0087 0.0113 0.0090 0.0155 0.0126
0.0213 0.0172 0.0093 0.0041 0.0041
0.0006 0.0041 0.0122 0.0385 0.0529 0.0647 0.0786 0.0997 0.0857 0.0997 0.1013 0.0818 0.0500
0.0479 0.0527 0.0263 0.0206 0.0255 0.0082 0.0128 0.0092 0.0036 0.0035 0.0067 0.0046
0.0004 0.0054 0.0026 0.0004 0.0000

####Number and vector of years of age compositions for commercial handline fleet
13
1997 1998 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
#Sample size of age comp data (first row observed Ntrips, second row Nfish)
11.0 12.0 9.0 7.0 18.0 49.0 36.0 110.0 101.0 115.0 111.0 89.0 113.0
105.0 72.0 60.0 83.0 215.0 381.0 189.0 963.0 538.0 455.0 735.0 599.0 834.0
#commercial handline age comps
0.0000 0.0477 0.1340 0.2162 0.2988 0.1062 0.0769 0.0538 0.0568 0.0071 0.0000 0.0000 0.0000
0.0024
0.0000 0.0667 0.2264 0.2145 0.2531 0.0870 0.1140 0.0000 0.0053 0.0249 0.0000 0.0000 0.0082
0.0000
0.0000 0.1055 0.3203 0.2042 0.2146 0.0497 0.0401 0.0397 0.0000 0.0000 0.0143 0.0042 0.0048
0.0026
0.0000 0.0919 0.2518 0.3007 0.2661 0.0634 0.0103 0.0041 0.0000 0.0117 0.0000 0.0000 0.0000
0.0000
0.0000 0.0487 0.1532 0.2362 0.2194 0.1451 0.0991 0.0513 0.0088 0.0075 0.0106 0.0052 0.0055
0.0094
0.0000 0.0017 0.0518 0.1450 0.2470 0.1984 0.1015 0.0796 0.0599 0.0456 0.0123 0.0208 0.0072
0.0291
0.0055 0.0128 0.0703 0.2032 0.3260 0.2271 0.0772 0.0384 0.0083 0.0208 0.0104 0.0000 0.0000
0.0000
0.0027 0.0245 0.0924 0.1139 0.1736 0.2481 0.1684 0.0899 0.0432 0.0176 0.0094 0.0091 0.0000
0.0072
0.0044 0.0232 0.0608 0.1276 0.1720 0.1762 0.1899 0.1005 0.0359 0.0449 0.0008 0.0172 0.0000
0.0466
0.0097 0.0685 0.1042 0.1334 0.0990 0.1486 0.1893 0.0735 0.0418 0.0265 0.0406 0.0023 0.0211
0.0415
0.0279 0.1856 0.2237 0.1336 0.1099 0.0606 0.0898 0.0687 0.0445 0.0204 0.0087 0.0012 0.0006
0.0249
0.0104 0.1462 0.2709 0.2250 0.1682 0.1027 0.0316 0.0190 0.0148 0.0104 0.0004 0.0000 0.0000
0.0004
0.0173 0.0786 0.2130 0.2447 0.1399 0.1234 0.0648 0.0408 0.0303 0.0218 0.0168 0.0000 0.0057
0.0021

#####################################Commercial Longline ##
##Starting and ending years for landings time series
1978
2012
#commercial longline landings vector (1000 lb whole weight) and assumed CVs (landings + discards)
45.87 41.96 42.74 47.16 103.69 323.41 225.40 149.22 171.11 183.70 153.10 191.68 227.53
154.20 226.73 196.88 109.67 97.66 64.46 174.29 84.71 91.30 100.10 43.01 27.09
22.70 53.86 36.32 42.58 3.70 10.85 7.93 3.08 1.26 2.70
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05

#Number and vector of years of commercial length compositions
14
1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996
1997
#Sample size of length comp data (first row observed Ntrips, second row Nfish)
24.0 28.0 19.0 14.0 14.0 7.0 15.0 22.0 63.0 100.0 41.0 48.0 18.0
33.0
1139.0 1065.0 1286.0 565.0 461.0 341.0 714.0 917.0 1700.0 4668.0 807.0 1755.0 757.0
1355.0
#commercial longline length comps (3 cm length bins)
0.0000 0.0007 0.0007 0.0021 0.0079 0.0123 0.0148 0.0185 0.0325 0.0365 0.0233 0.0148 0.0265
0.0258 0.0475 0.0415 0.0648 0.0540 0.0745 0.0677 0.0676 0.0608 0.0665 0.0630 0.0701
0.0537 0.0262 0.0178 0.0052 0.0027
0.0000 0.0000 0.0000 0.0047 0.0155 0.0289 0.0341 0.0476 0.0574 0.0580 0.0767 0.0781 0.0701
0.0835 0.0817 0.0610 0.0644 0.0473 0.0466 0.0291 0.0298 0.0246 0.0210 0.0102 0.0127
0.0096 0.0066 0.0007 0.0000 0.0000
0.0000 0.0000 0.0009 0.0035 0.0122 0.0197 0.0332 0.0402 0.0302 0.0267 0.0599 0.0319 0.0832
0.1094 0.0504 0.1019 0.0749 0.0421 0.0762 0.0390 0.0490 0.0149 0.0171 0.0333 0.0289
0.0131 0.0061 0.0013 0.0009 0.0000
0.0000 0.0000 0.0012 0.0012 0.0180 0.0290 0.0673 0.0583 0.0480 0.0738 0.0335 0.0547 0.0939
0.0311 0.0289 0.0323 0.0521 0.0435 0.0261 0.0463 0.0380 0.0535 0.0301 0.0478 0.0180
0.0210 0.0187 0.0140 0.0082 0.0117
0.0000 0.0044 0.0000 0.0419 0.0562 0.0896 0.1042 0.1204 0.1129 0.0880 0.1202 0.0916 0.0419
0.0131 0.0208 0.0269 0.0155 0.0027 0.0017 0.0068 0.0051 0.0094 0.0051 0.0087 0.0094
0.0017 0.0000 0.0000 0.0000 0.0017
0.0000 0.0000 0.0000 0.0000 0.0033 0.0025 0.0082 0.0090 0.0098 0.0627 0.0731 0.1090 0.1186
0.1098 0.1232 0.0758 0.0611 0.0158 0.0324 0.0420 0.0450 0.0485 0.0112 0.0066 0.0128
0.0090 0.0082 0.0016 0.0008 0.0000
0.0000 0.0000 0.0027 0.0231 0.0128 0.0297 0.0314 0.0620 0.0759 0.0665 0.0989 0.0616 0.0811
0.0789 0.0860 0.0544 0.0279 0.0081 0.0149 0.0252 0.0101 0.0276 0.0247 0.0296 0.0209
0.0238 0.0146 0.0063 0.0009 0.0005
0.0000 0.0000 0.0010 0.0061 0.0081 0.0329 0.0380 0.0729 0.0888 0.0741 0.1034 0.0856 0.0806
0.0759 0.0663 0.0582 0.0299 0.0268 0.0122 0.0152 0.0071 0.0117 0.0122 0.0142 0.0239
0.0163 0.0224 0.0056 0.0061 0.0046
0.0000 0.0000 0.0000 0.0014 0.0059 0.0161 0.0272 0.0289 0.0617 0.0772 0.0798 0.0762 0.1009
0.1040 0.0914 0.0680 0.0537 0.0438 0.0324 0.0224 0.0128 0.0140 0.0145 0.0127 0.0116
0.0131 0.0121 0.0110 0.0042 0.0032
0.0000 0.0000 0.0003 0.0038 0.0086 0.0258 0.0358 0.0629 0.0755 0.0812 0.0864 0.1002 0.1091
0.0906 0.0726 0.0656 0.0592 0.0309 0.0238 0.0134 0.0100 0.0070 0.0047 0.0075 0.0073
0.0062 0.0050 0.0049 0.0011 0.0005
0.0000 0.0013 0.0037 0.0183 0.0195 0.0381 0.0506 0.0591 0.0770 0.0753 0.0865 0.1020 0.0770
0.0746 0.0812 0.0795 0.0496 0.0285 0.0185 0.0161 0.0039 0.0136 0.0063 0.0025 0.0038
0.0062 0.0013 0.0013 0.0024 0.0024
0.0000 0.0000 0.0000 0.0094 0.0187 0.0435 0.0833 0.1227 0.1286 0.1366 0.0956 0.0826 0.0728
0.0417 0.0449 0.0446 0.0242 0.0182 0.0079 0.0072 0.0034 0.0007 0.0033 0.0007 0.0023
0.0059 0.0007 0.0000 0.0000 0.0007
0.0000 0.0025 0.0012 0.0135 0.0112 0.0504 0.0998 0.1332 0.1334 0.1382 0.1188 0.0913 0.0558
0.0468 0.0340 0.0161 0.0204 0.0069 0.0077 0.0039 0.0028 0.0041 0.0026 0.0026 0.0000
0.0014 0.0012 0.0000 0.0000 0.0000
0.0000 0.0006 0.0017 0.0081 0.0195 0.0213 0.0391 0.0806 0.1229 0.1327 0.1445 0.0992 0.0733
0.0666 0.0341 0.0639 0.0186 0.0168 0.0091 0.0071 0.0062 0.0238 0.0034 0.0006 0.0011
0.0017 0.0023 0.0006 0.0006 0.0000

#Number and vector of years of commercial age compositions
14
1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
2012
#Sample size of age comp data (first row observed Ntrips, second row Nfish)
7.0 6.0 11.0 10.0 9.0 6.0 5.0 6.0 19.0 9.0 12.0 21.0 13.0
10.0
62.0 64.0 109.0 104.0 117.0 69.0 86.0 41.0 161.0 33.0 53.0 51.0 35.0
44.0
#commercial longline age comps
0.0000 0.0531 0.1194 0.2077 0.2358 0.0789 0.1607 0.0635 0.0152 0.0386 0.0053 0.0113 0.0000
0.0106
0.0000 0.0000 0.0808 0.1656 0.2258 0.1986 0.1471 0.0829 0.0298 0.0275 0.0075 0.0000 0.0000
0.0344
0.0000 0.0000 0.0401 0.1568 0.1562 0.1300 0.1982 0.1525 0.0462 0.0337 0.0227 0.0099 0.0394
0.0141
0.0000 0.0028 0.0491 0.0454 0.0936 0.1804 0.2116 0.1202 0.0847 0.1055 0.0435 0.0303 0.0213
0.0086
0.0000 0.0525 0.0671 0.2551 0.1662 0.1235 0.1494 0.0451 0.0612 0.0116 0.0072 0.0220 0.0072
0.0317
0.0000 0.0000 0.1743 0.3619 0.2673 0.0895 0.0692 0.0377 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0885 0.2005 0.2792 0.2730 0.0628 0.0551 0.0282 0.0128 0.0000 0.0000 0.0000 0.0000
0.0000
0.0000 0.0685 0.1028 0.2486 0.3199 0.0560 0.0813 0.0703 0.0000 0.0348 0.0177 0.0000 0.0000
0.0000
0.0000 0.0053 0.0849 0.3265 0.2859 0.1410 0.0482 0.0328 0.0146 0.0031 0.0022 0.0000 0.0000
0.0556
0.0000 0.0000 0.0000 0.1715 0.3236 0.0971 0.1165 0.1553 0.0777 0.0000 0.0000 0.0000 0.0194
0.0388
0.0000 0.0000 0.0364 0.0354 0.1557 0.1883 0.2227 0.1283 0.1418 0.0000 0.0175 0.0000 0.0091
0.0648
0.0000 0.0000 0.0000 0.0000 0.1197 0.0241 0.1217 0.1616 0.2169 0.0119 0.0626 0.0846 0.0787
0.1182
0.0000 0.0000 0.0000 0.0536 0.0811 0.0728 0.1826 0.1122 0.0829 0.1122 0.0878 0.0978 0.0342
0.0829
0.0000 0.0000 0.1252 0.0461 0.0692 0.0263 0.0263 0.1414 0.0789 0.0690 0.0724 0.1117 0.0854
0.1479

```
###-><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
###-- BAM DATA SECTION: parameter section
###-><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
#
##################Parameter values and initial guesses#########################################################################################
#####################################################
###prior PDF (1=none, 2=lognormal, 3=normal, 4=beta)
#############################################################
##initial # lower # upper #       #  prior  # prior   # prior #
## guess  # bound # bound # phase #  mean   # var/-CV #  PDF  #
##--------#-------#-------#-------#---------#---------#------#
#                                        ###### Biological input ##################################
 1064.6     600.0   1500.0  -3     1064.6   -0.06      1     # VonBert Linf (units in mm TL)
 0.094      0.03    0.5     -3     0.094    -0.22      1     # VonBert K (units in mm TL)
-2.88      -4.0    0.0     -3    -2.88     -0.33      1     # VonBert t0 (units in mm TL)
 0.127      0.05    0.4      4     0.127    -0.13      3     # CV of length at age
 0.12       0.02    0.5     -3     0.12      0.016     1     # constant M (used only to compute MSST=(1-M)SSBmsy)
#                                        ###### SR parameters ######################################
 0.84       0.21    0.99    -3     0.84      0.0225    1     # SR steepness parameter
 #0.75      0.21    0.99     3     0.75      0.0196    4     # SR steepness parameter
 13.0       10.0    16.0     1     13.0      -0.5      1     # SR log_R0 parameter
 0.0       -1.0     1.0     -3     0.0       -0.5      1     # SR recruitment autocorrelation (lag 1)
 0.6        0.2     1.2      4     0.6       -0.25     3     # s.d. of recruitment in log space
#                                        ###### Selectivity parameters ##############################
 2.5        1.0     6.0      2     2.5       -0.5      3     # MMcvt age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # MMcvt slope of ascending limb
 6.0        2.0     10.0    -2     6.0       -0.5      1     # MMcvt age at peak selex = 1
 8.0        0.1     20.0     3     8.0       -0.5      3     # MMcvt descending limb (rate of descent)

 3.5        1.0     8.0      2     3.5       -0.5      3     # MMvll age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # MMvll longline slope of ascending limb

 3.5        1.0     7.0      2     3.5       -0.5      3     # comm handline age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # comm handline slope of ascending limb
 #5.0       2.0     10.0    -2     5.0       -0.5      1     # comm handline age at peak selex = 1
 #190.0     0.1     200.0    -3     30.0      -0.5      1     # comm handline descending limb (rate of descent)

 3.5        1.0     8.0      2     3.5       -0.5      3     # comm longline age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # comm longline slope of ascending limb

 3.0        1.0     7.0      2     3.0       -0.5      3     # BLOCK 1: recreational age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # recreational slope of ascending limb
 11.0       2.0     14.0    -2     10.0      -0.5      1     # recreational age at peak selex = 1
 10.0       0.1     20.0     3     10.0      -0.5      3     # recreational descending limb (rate of descent)

 3.0        0.5     5.0      2     3.0       -0.5      3     # BLOCK 2: recreational age at 50% selectivity
 2.0        0.01    10.0     2     2.0       -0.5      3     # recreational slope of ascending limb
 6.0        2.0     10.0    -2     5.0       -0.5      1     # recreational age at peak selex = 1
 8.0        0.1     20.0     3     8.0       -0.5      3     # recreational descending limb (rate of descent)

 3.0        0.5     5.0      2     3.0       -0.5      3     # BLOCK 3: recreational age at 50% selectivity
 2.0        0.1     10.0     2     2.0       -0.5      3     # recreational slope of ascending limb
 8.0        2.0     10.0    -2     6.0       -0.5      1     # recreational age at peak selex = 1
 8.0        0.1     20.0     3     8.0       -0.5      3     # recreational descending limb (rate of descent)
                                        ###### Index catchability parameters ########################
-10.0   -14.0    -3.0      1   -10.0      -0.5      1     # MMcvt CPUE (log q)
-10.0   -14.0    -3.0      1   -10.0      -0.5      1     # MMvll CPUE (log q)
-10.0   -14.0    -3.0      1   -10.0      -0.5      1     # HB CPUE (log q)
-8.0    -14.0    -3.0     -1    -7.0      -0.5      1     # comm handline CPUE (log q)
                                        ###### Fishing mortality parameters #########################
 0.03       0.0     0.2     -4     0.03      -0.5      1     # F used to initialize popn, distributed among fleets in proportion to their early Fs
-2.0       -8.0    1.0      1    -2.0       -0.5      1     # comm handlines log mean F
-2.0       -8.0    1.0      1    -2.0       -0.5      1     # comm longlines log mean F
-6.0       -10.0   1.0      1    -6.0       -0.5      1     # recreational log mean F

##### Dev vectors ###############################################################################
########################
# lower # upper #       #
# bound # bound # phase #
#-------#-------#-------#
 -10.0     5.0     2    # recreational F devs
 -10.0     5.0     2    # comm handline F devs
 -10.0     5.0     2    # comm handline F devs
 -5.0      5.0     2    # recruitment devs
 -5.0      5.0     1    # Nage devs

# rec F dev initial guesses
0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0
# commercial handline F dev initial guesses
0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0
# commercial longline F dev initial guesses
0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
# rec devs
0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
                0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0             0.0
```

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0
#initial N age devs, all ages but the first one
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: likelihood weights section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

##################Likelihood Component Weighting###
1.0 #landings

0.47 #MM cvt index
0.71 #MM vll index
0.50 #HB index
0.0 #comm handline index

1.18 #HB len comps
0.24 #cH len comps
0.45 #cL len comps

0.76 #MM cvt age comps
0.37 #MM vll age comps
0.29 #HB age comps
0.15 #cH age comps
0.39 #cL age comps

1.0 #log N.age.dev residuals (initial abundance)
1.0 #S-R residuals
1.0 #constraint on early recruitment deviations
0.0 #constraint on ending recruitment deviations
0.0 #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization) fULL F summed over fisheries
0.0 #weight on tuning F (penalty not applied in final phase of optimization)

##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##-- BAM DATA SECTION: miscellaneous stuff section
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

#length-weight (TL-whole wgt) coefficients a and b, W=aL^b, (W in kg, TL in mm)--sexes combined
4.63E-8
2.824

#time-invariant vector of % maturity-at-age for females (ages 1-25+)
0.000 0.000 0.133 0.239 0.391 0.568 0.729 0.846 0.919 0.959 0.979 0.990 0.995
0.998 0.999 0.999 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
#time-invariant vector of % maturity-at-age for males (ages 1-25+)
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 1.0 1.0 1.0 1.0 1.0
#time-invariant vector of proportion female (ages 1-25+)
1.000 1.000 1.000 1.000 0.978 0.967 0.953 0.934 0.910 0.879 0.842 0.798 0.748
0.692 0.631 0.566 0.499 0.433 0.368 0.307 0.251 0.200 0.157 0.120 0.027
#time of year (as fraction) for spawning: end of June (Wyanski et al. (99) found spawning season April - Sept.
0.5

#age-dependent natural mortality at age (ages 1-25+) -- scaled Charnov
0.439 0.344 0.284 0.243 0.214 0.192 0.175 0.161 0.151 0.142 0.135 0.129 0.123
0.119 0.115 0.112 0.109 0.107 0.104 0.102 0.101 0.099 0.098 0.097 0.096
#Max observed age
35

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

#rate increase switch: Integer value (choose estimation phase, negative value turns it off)
-1
#annual positive rate of increase on all fishery dependent q's due to technology creep
0.0
DD q switch: Integer value (choose estimation phase, negative value turns it off)
-1
#density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)
0.0
#SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)
0.128
#Age to begin counting D-D q (should be age near full exploitation)
5.0
#Random walk switch:Integer value (choose estimation phase, negative value turns it off)
-3
#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence
0.03

#Tuning F (not applied in last phase of optimization, or not applied at all if penalty weight=0)
0.3
#Year for tuning F
2000

#threshold sample sizes ntrips (>=)for length comps (set to 99999.0 if sel is fixed):
#applied nfish >=26 to data input
5.0 #rec
5.0 #comm cH
5.0 #comm cL

#threshold sample sizes ntrips (greater than or equal to) for age comps (set to 99999.0 if sel is fixed)
#applied nfish >=26 to data input
5.0 #MM cvt
5.0 #MM vll
5.0 #rec
5.0 #comm cH
5.0 #comm cL

#Ageing error matrix (columns are true age 1-25+, rows are ages as read for age comps: columns should sum to one)

1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

999 #end of data file flag