Technical documentation of the 
Beaufort Assessment Model (BAM)

Erik H. Williams
Kyle W. Shertzer

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Marine Fisheries Service
Southeast Fisheries Science Center
NOAA Beaufort Laboratory
101 Pivers Island Road
Beaufort, North Carolina 28516

February, 2015
Technical documentation of the Beaufort Assessment Model (BAM)

Erik H. Williams
Kyle W. Shertzer
Southeast Fisheries Science Center
Beaufort, North Carolina

February, 2015

This Technical Memorandum series is used for documentation and timely communication of preliminary results, interim reports, or similar special-purpose information. Although the memoranda are not subject to complete formal review, editorial control, or detailed editing, they are expected to reflect sound professional work.
NOTICE

The National Marine Fisheries Service (NMFS) does not approve, recommend or endorse any proprietary product or material mentioned in this publication. No reference shall be made to NMFS, or to this publication furnished by NMFS, in any advertising or sales promotion which would imply that NMFS approves, recommends, or endorses any proprietary product or proprietary material mentioned herein which has as its purpose any intent to cause directly or indirectly the advertised product to be used or purchased because of this NMFS publication.

This report should be cited as follows:


Copies may be obtained from:

National Technical Information Center
5825 Port Royal Road
Springfield, VA 22161
(800) 553-6842 or (703) 605-6000
http://www.ntis.gov/numbers.htm

or by contacting either author
Erik.Williams@noaa.gov or Kyle.Shertzer@noaa.gov

PDF version available at http://www.sefsc.noaa.gov/
1 Overview

The Beaufort Assessment Model (BAM) applies a statistical catch-age formulation implemented with the AD Model Builder software (Fournier et al. 2012) and fitted to multiple data sources simultaneously in a single integrated analysis (Maunder and Punt 2013). In essence, the model simulates a population forward in time while including fishing and biological processes (Quinn and Deriso 1999; Shertzer et al. 2014). Quantities to be estimated are systematically varied until characteristics of the simulated population match available data on the real population. Its basic structure is similar to that of other packages such as Stock Synthesis (Methot 2012; Methot and Wetzel 2013) and Age Structured Assessment Program (Legault 2008).

Simulation testing has shown that the BAM can recover estimated parameters accurately. Furthermore, the code and general model structure have been implemented by multiple analysts and have been through numerous independent reviews. Versions of BAM have been applied in peer-reviewed publications [e.g., Conn et al. (2010)] and in stock assessments of Atlantic menhaden (Brevoortia tyrannus), Gulf menhaden (Brevoortia patronus), Spanish mackerel (Scomberomorus maculatus), and numerous reef fishes off the southeast U.S. coast, such as black sea bass (Centropristis striata), blue-line tilefish (Caulolatilus microps), gag (Mycteroperca microlepis), greater amberjack (Seriola dumerili), red grouper (Epinephelus morio), red porgy (Pagrus pagrus), red snapper (Lutjanus campechanus), snowy grouper (Hyporthodus nivosus or Epinephelus nivosus), tilefish (Lopholatilus chamaeleonticeps), and vermilion snapper (Rhomboplites aurorubens). Assessment reports are available at http://www.sefsc.noaa.gov/sedar.

2 Model description

BAM is fundamentally an age-structured population model with birth and death processes. New biomass is acquired through growth and recruitment, while abundance of existing cohorts experiences exponential decay from fishing and natural mortality. The population is assumed closed to immigration and emigration. The model follows an annual time step for \( n \) years, \( y_1, \ldots, y_n \), and it includes \( A \) age classes \( 1^\ldots A^+ \), where the oldest age class \( A^+ \) allows for the accumulation of fish (i.e., plus group). The youngest age class (recruits) is typically age-1 fish produced by the previous year’s spawners, but it could instead be age-0 fish produced by the current year’s spawners (and consequently with \( A+1 \) age classes). Subsequent descriptions assume age-1 is the youngest age class.

Model notation and details are described below and in Table 1, and the basic flow of operations is illustrated in Fig. 1. Although some features of the source code are generalized, others are customized to each stock assessment. Thus, application of BAM requires some programming of the AD Model Builder template file (i.e., \( \text{filename}.tpl \)), as well as configuration of the data input file (i.e., \( \text{filename}.dat \)). This has its drawbacks, notably that application of BAM requires a working knowledge of AD Model Builder and user effort to code the tpl file. It also has its benefits, primarily that the model configuration is extremely flexible, which allows maximum customization for any particular stock assessment and relatively quick modification if needed. This latter ability can be quite useful during stock assessment workshops. In addition, BAM is not static but continues to evolve as the field of stock assessment advances. Thus, because of BAM’s flexibility and continued development, the description below is intended as a general documentation of model structure, not an exhaustive account of all possible features. An example application using gag is provided in the Appendix.

2.1 Initialization

BAM has several options to compute initial abundance at age, i.e., abundance in the first modeled year. In all cases, the equilibrium age structure is computed based on natural and initial fishing mortality \( (F_{\text{init}}) \), where \( F_{\text{init}} \) is typically defined in one of three ways: 1) input as a fixed value, 2) assumed equal to the average \( F \) from the first few years (usually three) of the assessment, and 3) estimated, either freely or with a prior, as its own parameter or as a proportion of the average \( F \) from the first few years of the assessment. In some assessments, the equilibrium age structure is used to initialize the population. However, other assessments attempt to estimate the initial nonequilibrium age structure, if composition data are available to inform these estimates. If so, estimation follows a two-part procedure, where first the equilibrium age...
structure is computed as described above, and second, lognormal deviations \( \hat{\sigma}_{a}^{\text{init}} \) around the equilibrium are estimated for each age two and older. The deviations are penalized by the squared Euclidean norm function, i.e., sum of squares. Consequently, the initial abundance of each age can vary from equilibrium if suggested by early composition data, but remain estimable if data are uninformative. The initial spawning stock, computed from the initial abundance of ages \( 2^+ \), is used to generate the number of recruits (age-1 fish) in the first assessment year, using methods similar to those in subsequent years (described below).

2.2 Growth of individuals

Mean length \((l_a, \text{in units of mm})\) at age of the population is modeled with the von Bertalanffy equation,

\[
l_a = L_\infty (1 - \exp[-K(a - t_0 + \tau)])
\]

where \( L_\infty \), \( K \), and \( t_0 \) are parameters, and \( \tau \) is a fixed value to represent a fraction of the year (typically \( \tau = 0.5 \)). In some assessments, these parameters are estimated within the assessment model, often informed by prior distributions. In other assessments, the parameters are estimated from data before the assessment and treated by BAM as input. Variation in length at age is assumed to be normally distributed, with a CV or standard deviation that is typically estimated and assumed constant across ages, but can also be configured to vary with age.

Weight at age \((w_a, \text{in kg of whole weight, WW})\) is treated as input or else modeled as a function of mean length. If modeled, the functional form is specified by the user, commonly as a power function, \( w_a = \theta_1 l_a^{\theta_2} \), where \( \theta_1 \) and \( \theta_2 \) are parameters. These parameters are typically estimated from data before the assessment and treated by BAM as input. Once whole weight at age in kg is computed, various conversions may be applied as needed, for example from kilograms to metric tons or to pounds, or from whole weight to gutted weight (GW).

In some cases, fishing fleets might target fish of different sizes than those in the population at large. If so, length at age would differ from \( l_a \) in Equation 1 (Schueller et al. 2014). The BAM accommodates this feature by allowing for fleet-specific growth curves, which would translate into fleet-specific weights at age.

2.3 Natural mortality rate

The natural mortality rate \((M)\) is typically treated as input, but in some cases can be estimated. The form of \( M \) as a function of age is defined by the user. It could be specified as constant (i.e., age independent), but more commonly \( M \) is assumed to decrease with age or size. For assessments in the southeast U.S., age-specific \( M \) has typically been based on studies by Lorenzen (1996) or, more recently, Charnov et al. (2013). The Lorenzen (1996) approach inversely relates the natural mortality at age to mean weight at age \( W_a \) by the power function \( M_a = \alpha W_a^\beta \), where \( \alpha \) is a scale parameter and \( \beta \) is a shape parameter. Lorenzen (1996) provided point estimates of \( \hat{\alpha} = 3.69 \) and \( \hat{\beta} = -0.305 \) for oceanic fishes. Similarly, the Charnov et al. (2013) approach inversely relates the natural mortality at age to somatic growth, \( M_a = K(l_a/L_\infty)^{-1.5} \). Whichever approach is taken, the age-dependent estimates of \( M_a \) are often rescaled for consistency with cumulative survival to maximum age (Hoenig 1983; Hewitt and Hoenig 2005; Then et al. 2014). In some assessments, \( M_a \) is assumed to vary across years.

2.4 Maturity and sex ratio

Maturity at age of females is treated by BAM as input, either as a vector (i.e., if time invariant) or an \( n \times A \) matrix (i.e., for year- and age-specific values). Sex ratio at age is treated in the same manner. Many stocks in the southeast U.S. are protogynous hermaphrodites, and so for those stocks, maturity at age of males is also modeled.
2.5 Reproductive output

BAM is flexible in how it computes reproductive output, often referred to as spawning stock \((S)\). For gonochoristic species, reproductive output is typically computed as total fecundity (when that information is available), or else as mature female biomass (in units of mt). For protogynous species, reproductive output is typically modeled as total mature biomass (mt; males and females), following the advice of Brooks et al. (2008). Computations discount abundance to the time of peak spawning, thus accounting for a partial year of natural and fishing mortality.

2.6 Recruitment

Expected annual recruitment \((\bar{R}_y)\) is computed from either the Beverton–Holt or Ricker spawner-recruit model. In BAM, the Beverton–Holt formulation is,

\[
\bar{R}_{y+1} = \frac{0.8R_0hS_y}{0.2R_0\phi_0(1-h) + S_y(h-0.2)}
\]

(2)

where \(R_0\) is virgin recruitment, \(h\) is steepness, and \(\phi_0\) is the unfished spawners per recruit. The analogous Ricker formulation is,

\[
\bar{R}_{y+1} = \frac{S_y}{\phi_0} \exp \left( h \left( 1 - \frac{S_y}{R_0\phi_0} \right) \right)
\]

(3)

In years when data are considered to be informative on recruitment, multiplicative deviations are included assuming a lognormal distribution,

\[
N_{1,y} = \bar{R}_y \exp(r_y)
\]

(4)

Here \(r_y\) is assumed to follow a normal distribution with standard deviation \(\sigma_R\).

In arithmetic space, expected recruitment is higher than that estimated directly from the spawner-recruit curve because of lognormal deviation in recruitment residuals. Thus, a bias correction is applied when computing equilibrium recruitment. The bias correction \((\varsigma)\) is computed from the variance \((\sigma^2_R)\) of recruitment deviations in log space: \(\varsigma = \exp(\sigma^2_R/2)\). Then, under Beverton–Holt, the expected equilibrium recruitment \((R^{eq})\) associated with any \(F\) is,

\[
R^{eq} = \frac{R_0[\varsigma h \phi_F - (1-h)\phi_0]}{(5h-1)\phi_F}
\]

(5)

and under Ricker,

\[
R^{eq} = \frac{R_0}{\phi_F} \left( 1 + \frac{\log(\varsigma \Phi_F)}{h} \right)
\]

(6)

where \(\phi_F\) is spawners per recruit given \(F\), and \(\Phi_F = \phi_F/\phi_0\) is the spawning potential ratio.

In years when data are considered to be uninformative on recruitment, multiplicative deviations would not generally be estimated. Instead, \(N_{1,y+1} = R^{eq}\). Computation of \(R^{eq}\), along with the mortality schedule, implies an equilibrium age structure, which would apply to calculations of the initialization (described above) as well as calculations of biological reference points (described below).
2.7 Selectivities

In BAM, selectivity is modeled as a function of age. It may also vary over time, but to simplify the description below, the year subscript is not included. Selectivity at age \( s(f,d,u,a) \) ranges on the interval \([0, 1]\) and can be modeled for three different types of data: landings (denoted by subscript \( f \)), discards (subscript \( d \)), and indices (subscript \( u \)). In any case, it may be estimated by using a free parameter \( x(f,d,u,a) \) for each age, or by using a parametric function. The free-parameter approach estimates selectivity in logit space, such that \( s(f,d,u,a) = \frac{1}{1+\exp(-x(f,d,u,a))} \).

The parametric approach imposes theoretical structure on selectivity, and it can reduce the number of estimated parameters, particularly when the model includes many ages. Parametric models of selectivity in BAM impose one of two forms: flat-topped or dome-shaped. Flat-topped selectivity describes a pattern of fishing rates that increase across the younger ages and then saturate at a value of 1.0 for all older ages. In BAM, it is estimated using a two-parameter \( (x_1, x_2) \) logistic model:

\[
s(f,d,u,a) = \frac{1}{1 + \exp(-x_1(a - x_2))}
\]

where \( x_1 \) controls the rate of increase, and \( x_2 \) is the age at 50% selection.

Dome-shaped selectivity describes a pattern of fishing rates that increase across the younger ages, peak at a value of 1.0, and then decrease across older ages. In BAM, four options are available for dome-shaped selectivity: double-logistic, joint-logistic, logistic-exponential, and double-Gaussian. The double-logistic model (four parameters) combines two logistic curves, one to describe the increasing portion and one to describe the decreasing portion:

\[
s(f,d,u,a) = \frac{1}{1 + \exp(-x_1(a - x_2))} \left( \frac{1}{1 + \exp(-x_3(a - x_4))} \right)
\]

The double-logistic model typically requires re-scaling to ensure that it peaks at one. As such, parameters may not be identifiable without the use of priors.

The joint-logistic model (five parameters) does not require re-scaling, but does require specifying \textit{a priori} the age at full selection \( (a_f) \). In addition, this model allows the descending limb to saturate at a value \( (x_5) \) less than 1.0:

\[
s(f,d,u,a) = \begin{cases} 
1 + \exp(-x_1(a - x_2)) & : \ a < a_f \\
1 & : \ a = a_f \\
1 - \frac{1-x_5}{1+\exp(-x_3(a - x_4))} & : \ a > a_f
\end{cases}
\]

Similarly, the logistic-exponential model (three parameters) requires specifying \textit{a priori} the age at full selection. It describes the ascending limb with a logistic curve for ages prior to full selection (two parameters \( x_1, x_2 \)), and the descending limb with a negative exponential curve (one parameter, \( x_3 \)):

\[
s(f,d,u,a) = \begin{cases} 
\frac{1}{1+\exp(-x_1(a-x_2))} & : \ a < a_f \\
1 & : \ a = a_f \\
\exp \left( -\left( \frac{(a-a_f)}{x_3} \right)^2 \right) & : \ a > a_f
\end{cases}
\]

The double-gaussian model (six parameters) is the most flexible option in BAM, but does require re-scaling. Parameters are loosely defined as follows: \( x_1' \) is the ascending inflection location, \( x_2' \) controls the width of the plateau, \( x_3' \) controls the ascent width, \( x_4' \) controls the descent width, \( x_5' \) controls the function value at the youngest age, and \( x_6' \) controls the

4
function value at the oldest age. These parameters are transformed as follow:

\[
\begin{align*}
x_1 &= x'_{1} \\
x_2 &= x'_{1} + 1.0 + \frac{(0.99A-x'_1-1.0)}{1+\exp(-x'_2)} \\
x_3 &= \exp(x'_3) \\
x_4 &= \exp(x'_4) \\
x_5 &= \frac{1.0}{1.0+\exp(-x'_5)} \\
x_6 &= \frac{1.0}{1.0+\exp(-x'_6)}
\end{align*}
\]

(11)

Given the transformed parameters, several intermediate functions are defined:

\[
\begin{align*}
f_1(a) &= \exp\left(-\frac{(a-x_1)^2}{x_3}\right) \\
f_2(a) &= x_5 + (1.0 + x_5)(\frac{f_1(a)-f_1(a_1)}{(1.0-f_1(a_1)}) \\
f_3(a) &= \exp\left(-\frac{(a-x_2)^2}{x_4}\right) \\
f_4(a) &= 1.0 + (x_6 - 1)(\frac{f_3(a)-1.0}{f_3(A)-1.0}) \\
f_5(a) &= \frac{1.0}{1.0+\exp\left(-\frac{20(a-x_1)}{1.0+|a-x_1|}\right)} \\
f_6(a) &= \frac{1.0}{1.0+\exp\left(-\frac{20(a-x_2)}{1.0+|a-x_2|}\right)}
\end{align*}
\]

(12)

Here, \(a_1\) is the youngest age (typically 0 or 1), and \(A\) is the oldest age. Then, using the intermediate functions, selectivity is computed as:

\[
s_{(f,d,u),a} = f_2(a) (1.0 + f_5(a)) + f_5(a) [1.0 - f_6(a) + f_4(a)f_6(a)]
\]

(13)

Whichever approach is used, selectivity functions may vary over time, and thus in practice have the additional subscript of year, \(s_{(f,d,u),a,y}\). The variation could be annual or across blocks of years, for example, where blocks represent periods of consistent regulations. Age and/or length composition data are critical to estimating selectivity, but even with those data, parameters will not always be identifiable without the use of priors.

### 2.8 Fishing

For each fleet being modeled, the BAM estimates a separate full fishing mortality rate for each year of the time series \(F_{(f,d),y}\), with landings and discards treated as distinct fleets. Age-specific rates are computed as the product of full \(F\) and selectivity at age (i.e., \(F_{(f,d),a,y} = s_{(f,d),a,y}F_{(f,d),y}\)). Then, the across-fleet annual \(F_y\) is represented by apical \(F\), computed as the maximum of \(F\) at age summed across fleets,

\[
F_{a,y} = \sum_{(f,d)} F_{(f,d),a,y}
\]

(14a)

\[
F_y = \max_a (F_{a,y})
\]

(14b)

### 2.9 Landings and discard mortality

Landings at age in numbers for each fleet are predicted using the Baranov catch equation (Baranov 1918),

\[
l'_{f,a,y} = \frac{F_{f,a,y}}{Z_{a,y}} N_{a,y} [1 - \exp(-Z_{a,y})]
\]

(15)
where \( Z_{a,y} = M_a + F_{a,y} \) is total mortality at age and \( N_{a,y} \) is annual abundance at age. Then, landings at age in weight are calculated as,

\[
l'_{f,a,y} = Cw_{f,a,y}l_{f,a,y}
\]

where \( w_{f,a,y} \) is fleet-specific weight at age, which may differ from that of the population at large. The constant \( C \) converts units from those of \( w \) to those of observed removals (e.g., from mt to 1000 lb). Total landings in numbers and weight are computed as,

\[
L'_{f,y} = \sum_a l'_{f,a,y}
\]

\[
L''_{f,y} = \sum_a l''_{f,a,y}
\]

Similarly, dead discards at age in numbers from each discard fleet are computed,

\[
d'_{d,a,y} = F_{d,a,y}Z_{a,y}N_{a,y}[1 - \exp(-Z_{a,y})]
\]

as are those in weight,

\[
d''_{d,a,y} = Cw_{d,a,y}d'_{d,a,y}
\]

Total discards in numbers and weight are computed as,

\[
D'_{f,y} = \sum_a d'_{d,a,y}
\]

\[
D''_{f,y} = \sum_a d''_{d,a,y}
\]

### 2.10 Stock dynamics

Abundance of recruits \((N_{1,y})\) is described above in the section titled Recruitment. Abundance of each subsequent age at the start of each year is computed assuming exponential decay,

\[
N_{a+1,y+1} = N_{a,y}\exp(-Z_{a,y}) \quad \forall a \in (1 \ldots A - 1)
\]

\[
N_{A,y+1} = N_{A-1,y}\exp(-Z_{A-1,y}) + N_{A,y}\exp(-Z_{A,y})
\]

In addition, BAM computes abundance later in the year, \( N'_{a,y} = N_{a,y}\exp(-t_{\text{index}}Z_{a,y}) \), for matching observed indices of abundance. In this calculation, \( t_{\text{index}} \) represents the fraction of the year over which to apply total mortality, most typically \( t_{\text{index}} = 0.5 \) for calculating mid-year abundance. Similarly, BAM computes abundance at the time of peak spawning, \( N''_{a,y} = N_{a,y}\exp(-t_{\text{spawn}}Z_{a,y}) \), to derive spawning stock. Here, \( t_{\text{spawn}} \) represents the fraction of the year when peak spawning occurs (e.g., \( t_{\text{spawn}} = 0.25 \) reflects peak spawning at the end of March).

### 2.11 Indices of abundance

Predicted indices \((U_{u,y})\) for each index \((u)\) are computed from numbers at age, scaled to the relevant portion of the age structure by selectivity. A predicted index could additionally be computed in weight, if the observed index is measured in weight.

\[
U_{u,y} = \begin{cases} 
\hat{q}_{u,y} \sum_a s_{u,a}N'_{a,y} & : \text{if in numbers} \\
\hat{q}_{u,y} \sum_a s_{u,a}w_{a}N_{a,y} & : \text{if in weight} 
\end{cases}
\]

Catchability \((q_{u,y})\) scales indices of abundance to the estimated population at large.
2.12 Catchability

Annual catchability associated with each index can be modeled as constant or variable through time. Constant catchability is often the default assumption, but when available data allow for meaningful estimation, modeling catchability as time-varying may be desirable (SEDAR Procedural Guidance 2009; Wilberg et al. 2010). In BAM, three types of time-varying catchability are included as options: 1) density dependent, 2) linearly increasing, and 3) penalized random walk. The three options operate multiplicatively, and can be applied in any combination.

Density dependence is applied via a function, \( f_{\text{density}}(B^t_y) = (B^t_0)^\psi(B^t_y) - \hat{\psi} \), where \( \hat{\psi} \) is a parameter to be estimated or fixed, \( B^t_y = \sum_{a=0}^{A} B_{a,y} \) is annual biomass above some threshold age \( a' \), and \( B^t_0 \) is unfished biomass for ages \( a' \) and older. In practice, \( a' \) should be set high enough to reflect the exploitable biomass.

A linearly increasing trend is applied via the function, \( f_{\text{trend}}(y) \), which is set to 1.0 in year one \( (y_{u,1}) \) of the index, and increases thereafter according to the slope \( B_q \): \( f_{\text{trend}}(y) = f_{\text{trend}}(y-1) \times (y - y_{u,1})B_q \). Several applications of BAM have applied a slope of 2% per year to account for technological improvements in fishing efficiency. This increasing trend reflects the belief that catchability has generally increased over time as a result of improved technology (SEDAR Procedural Guidance 2009) and as estimated for reef fishes in the Gulf of Mexico (Thorson and Berkson 2010).

A random walk is applied assuming lognormal deviations, \( f_{\text{rw}}(y) = \exp(\epsilon_{u,y}) \). The values, \( \epsilon_{u,y} \), are penalized for deviation from zero, as described below in §3. The amount of “tension” on the random walk is controlled by an input parameter, \( \sigma_u^a \). As \( \sigma_u^a \) decreases, variation in the random walk is penalized more heavily.

Any of the time-varying functions not in use can simply be set to a value of 1.0. Then, annual catchability is computed as the product of the scaling constant, \( \hat{q}_{u}^t \), and each of the functions,

\[
q_{u,y} = \hat{q}_{u}^t \times f_{\text{density}}(B^t_y) \times f_{\text{trend}}(y) \times f_{\text{rw}}(y) \tag{23}
\]

If time-varying catchability is not modeled, all of the functions are set to 1.0, such that \( q_{u,y} = \hat{q}_{u}^t \).

2.13 Ageing error

The BAM can accommodate ageing error through application of a \( B^\alpha \times B^\alpha \) matrix \( E \), where \( B^\alpha \) is the number of age classes. In this matrix, the columns sum to one and act to spread true ages across ages that would be observed given ageing error. Predicted age compositions incorporate \( E \) for matching observed age compositions, as described in Table 1. If ageing error is not included, the matrix is set equal to the identity matrix, \( E = I \).

3 Fitting criteria

The objective function minimized by AD Model Builder is a composite of negative likelihoods with some additional penalty terms. Observed landings \( (\hat{L}) \), discards \( (\hat{D}) \), and indices \( (\hat{U}) \) are fit using lognormal likelihoods. Observed age compositions \( (\hat{p}^\alpha) \) and length compositions \( (\hat{p}^\lambda) \) are fit using standard or robust multinomial likelihoods (Francis 2011). In addition, the objective function includes various penalties, applied for two reasons: 1) to include prior information on estimated parameters, as might be done in a Bayesian analysis, and 2) to constrain variability within estimated vectors, such as annual recruitment deviations, random walk in catchability, and initial age structure. Although BAM contains common formulations of likelihoods and penalties, the objective function can be customized by the user to include virtually any fitting criteria.
3.1 Data components

Observed landings can be supplied in numbers or in weight for any given fleet. For fitting landings data, BAM uses the corresponding prediction \( L \), computed such that units of predictions and observations match, i.e., \( L = L' \) or \( L = L'' \). The landings contribution \( \Lambda^L \) to the total objective function is

\[
\Lambda^L = \sum_f \sum_y \left[ \log \left( \frac{(L_{f,y} + \epsilon)/(L_{f,y} + \epsilon)}{2 \sigma_{f,y}^2} \right) \right]^2
\]

where \( \epsilon = 1.0 - 5 \) to prevent the optimization procedure from attempting to compute the log of zero (an undefined value), and where \( \sigma_{f,y}^L \) are standard deviations in log space. These standard deviations are computed as \( \sigma_{f,y}^L = \sqrt{\log(1 + (CV_{f,y}^L/\omega_f^L)^2)} \), where \( CV_{f,y}^L \) are user-supplied coefficients of variation in arithmetic space and \( \omega_f^L \) are user-supplied weights. Analogous contributions to the total objective function are computed for discards \( \Lambda^D \) and indices of abundance \( \Lambda^B \).

Composition data are typically fit using a robust formulation of the multinomial likelihood (Francis 2011). In this formulation, predicted age compositions \((\hat{p}_{f,u,a,y}^\alpha)\) of fleet \( f \) or index \( u \) are matched to the observed values \((\hat{p}_{f,u,a,y}^\alpha)\), with contribution \( \Lambda^\alpha \) to the total objective function computed as,

\[
\Lambda^\alpha = \sum_{f,u} \sum_y 0.5 \log(E') - \log \left[ \exp \left( -\frac{(\hat{p}_{f,u,a,y}^\alpha - p_{f,u,a,y})^2}{2E'/(n_{f,u,a,y}^\alpha \omega_{f,u,a,y}^\alpha)} \right) + \epsilon \right]
\]

where \( E' = \left[ (1 - \hat{p}_{f,u,a,y}^\alpha)(\hat{p}_{f,u,a,y}^\alpha + 0.1) \right] \), \( B^\alpha \) is the number of age bins, \( n_{f,u,a,y}^\alpha \) are sample sizes, \( \omega_{f,u,a,y}^\alpha \) are user-supplied weights, and \( \epsilon = 1.0 - 5 \) to avoid log zero. Analogous contributions to the total objective function are computed for length composition data \( \Lambda^L \). The standard formulation of the multinomial likelihood (i.e., not the robust version) is also available as an option.

3.2 Penalty terms

Recruitment deviations are assumed to follow a lognormal distribution, with the option to allow first-order autocorrelation,

\[
\Lambda^{R1} = \omega_{R1} \left[ \frac{[r_y - \hat{p}_y]^2}{2 \hat{\sigma}_y^2} \right] + \sum_{y'=y'}^{y''} \frac{([r_y - \hat{p}_{y-1}] + \hat{\sigma}_y^2/2)^2}{2 \hat{\sigma}_y^2} + n \log(\hat{\sigma}_R)
\]

where \( r_y \) are recruitment deviations in log space, \( n \) is the number of years, \( \omega_{R1} \) is a user-supplied weight (may be 1.0), \( \hat{p} \) is the autocorrelation term, and \( \hat{\sigma}_y^2 \) is the estimated recruitment variance. The years \( y' \) and \( y'' \) are the first and last years for estimating recruitment deviations, which need not be the first and last years of the full assessment period. BAM includes the option for early recruitment deviations to receive additional constraint through a sum-of-squares penalty, \( \Lambda^{R2} = \omega_{R2} \sum_y r_y^2 \), applied over years \( y \). This penalty can be turned off by setting \( \omega_{R2} = 0 \). Similarly, terminal recruitment deviations may receive additional constraint if desired, \( \Lambda^{R3} = \omega_{R3} \sum_y r_y^2 \), which can be turned off by setting \( \omega_{R3} = 0 \).

If a nonequilibrium initial age structure is estimated, the deviations \( \hat{\sigma}_a^{init} \) from equilibrium are assumed to be lognormally distributed. They are penalized for deviating from zero using a sum-of-squares term, \( \Lambda^{init} = \omega_{init} \sum_a (\hat{\sigma}_a^{init})^2 \). These deviations do not include the youngest age, because it is already accounted for by the first year of recruitment deviations.

Similarly, if a random walk is applied to the catchability of index \( u \), a sum-of-squares penalty is applied, \( \Lambda^u = \sum_u \sum_y (\epsilon_{u,y}^2)/(2\sigma_u^2) \). Here, \( \sigma_u^2 \) controls the amount of tension on each random walk.
BAM includes an option to penalize apical $F_y$ if it exceeds a threshold value $\phi$, which is set by the user. The penalty is zero if $F_y \leq \phi$ and otherwise grows exponentially,

$$\Lambda^F = \omega_F \sum_y (\exp(F_y - \phi) - 1) \quad \forall F_y > \phi$$

(27)

This penalty is turned off when the user-defined weight $\omega_F$ is set to zero.

For any estimated parameter, a penalty can be applied for deviation from a user-supplied value. These penalties are similar in concept to prior distributions used in Bayesian approaches. Their purpose in BAM is to maintain parameter estimates near reasonable values and to prevent the optimization routine from drifting into parameter space with negligible gradient in the likelihood. This prior information on any given parameter is implemented as a negative log-likelihood term using one of three standard distributional forms that the user must specify: normal, lognormal, or beta. In addition, the user must specify the mean and variance of each distribution. The sum of all such penalty terms (i.e., negative log-likelihoods) is labeled $\Lambda^P$.

Given the data components and penalty terms, the total objective function value to be minimized is,

$$\Lambda = \Lambda^L + \Lambda^D + \Lambda^U + \Lambda^\alpha + \Lambda^\lambda + \Lambda^{R1} + \Lambda^{R2} + \Lambda^{R3} + \Lambda^{init} + \Lambda^q + \Lambda^F + \Lambda^P$$

(28)

4 Biological reference points

Biological reference points (benchmarks) are calculated based on maximum sustainable yield (MSY) estimates from the spawner-recruit model with bias correction (expected values in arithmetic space). These benchmarks include MSY, fishing mortality rate at MSY ($F_{MSY}$), dead discards at MSY ($D_{MSY}$), and spawning stock at MSY ($SSB_{MSY}$). The point of maximum yield is identified from the spawner-recruit curve and parameters describing growth, natural mortality, maturity, and selectivity. The value of $F_{MSY}$ is the $F$ that maximizes equilibrium landings (i.e., MSY). The values of $D_{MSY}$ and $SSB_{MSY}$ are those that correspond to $F_{MSY}$.

In addition to the MSY-related benchmarks, the assessment considered proxies based on per recruit analyses (e.g., $F_{40\%}$). The values of $F_{X\%}$ are defined as those $F$s corresponding to $X\%$ spawning potential ratio, i.e., spawners (population fecundity) per recruit relative to that at the unfished level. These quantities may serve as proxies for $F_{MSY}$ if the spawner-recruit relationship cannot be estimated reliably. Mace (1994) recommended $F_{40\%}$ as a proxy; however, later studies have found that a fishing rate of $F_{40\%}$ is too high across many life-history strategies (Williams and Shertzer 2003; Brooks et al. 2009) and can lead to undesirably low levels of biomass and recruitment (Clark 2002).

The MSY-based benchmarks and proxies are conditional on the estimated selectivity functions. For computation of benchmarks, three composite selectivities are computed from the terminal year of the assessment: 1) selectivity associated with landings, 2) selectivity associated with dead discards, and 3) the sum of the previous two, which describes total fishing mortality and has a peak value of one. The composite selectivities are $F$-weighted average selectivities across fleets, with $F$ from each fleet estimated as the full $F$ averaged over the last $X$ years of the assessment. Typically, $X = 3$ years.

5 Acknowledgments

The BAM has benefited from the analytical scrutiny of Lew Coggins, Paul Conn, Kevin Craig, Mike Prager, Amy Schueller, and Katie Siegfried. The authors are grateful for helpful comments on this memorandum by Kevin Craig, Amy Schueller, and Katie Siegfried.
6 References

References


<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Description or definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Labels for Indexing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Years</td>
<td>$y$</td>
<td>$y \in {y_1 \ldots y_n}$.</td>
</tr>
<tr>
<td>Ages</td>
<td>$a$</td>
<td>$a \in {a_1 \ldots A}$, where $a_1$ is the recruitment age, typically 0 or 1, and $A$ is the oldest age, treated as a plus group. The number of age bins is denoted $B^a$; typically $B^a = A$ or $A + 1$.</td>
</tr>
<tr>
<td>Length bins</td>
<td>$l$</td>
<td>$l \in {1 \ldots B^l}$, where $B^l$ is the number of length bins.</td>
</tr>
<tr>
<td>Length bin boundaries</td>
<td>$l'$</td>
<td>$l' \in {l'<em>1 \ldots l'</em>{B^l}}$, with values representing the upper bound of each length bin. Largest length bin is treated as a plus group.</td>
</tr>
<tr>
<td>Fleets, landings</td>
<td>$f$</td>
<td>Various fleets from commercial and/or recreational sectors.</td>
</tr>
<tr>
<td>Fleets, discards</td>
<td>$d$</td>
<td>Various fleets from commercial and/or recreational sectors.</td>
</tr>
<tr>
<td>Indices of abundance</td>
<td>$u$</td>
<td>Fishery dependent and/or fishery independent sources.</td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observed length compositions</td>
<td>$\tilde{p}^l_{(f,d,u),l,y}$</td>
<td>Proportional contribution of length bin $l$ in year $y$ to fleet $f, d$ (landings, discards) or index $u$.</td>
</tr>
<tr>
<td>Observed age compositions</td>
<td>$\tilde{p}^a_{(f,d,u),a,y}$</td>
<td>Proportional contribution of age class $a$ in year $y$ to fleet $f, d$ or index $u$.</td>
</tr>
<tr>
<td>Length comp. sample sizes</td>
<td>$\tilde{n}^l_{(f,d,u),y}$</td>
<td>Effective number of length samples collected in year $y$ from fleet $f, d$, or index $u$.</td>
</tr>
<tr>
<td>Age comp. sample sizes</td>
<td>$\tilde{n}^a_{(f,d,u),y}$</td>
<td>Effective number of age samples collected in year $y$ from fleet $f, d$ or index $u$.</td>
</tr>
<tr>
<td>Observed landings</td>
<td>$\tilde{L}_{f,y}$</td>
<td>Reported landings in year $y$ from fleet $f$.</td>
</tr>
<tr>
<td>CVs of landings</td>
<td>$CV_{L_{f,y}}^L$</td>
<td>Annual CV in arithmetic space.</td>
</tr>
<tr>
<td>Observed abundance indices</td>
<td>$\tilde{U}_{u,y}$</td>
<td>Relative abundance in year $y$ from index $u$.</td>
</tr>
<tr>
<td>CVs of abundance indices</td>
<td>$CV_{U_{u,y}}^U$</td>
<td>Annual CV in arithmetic space.</td>
</tr>
<tr>
<td>Observed total discards</td>
<td>$\tilde{D}_{d,y}$</td>
<td>Reported total discards (live and dead) in year $y$ from fleet $d$.</td>
</tr>
<tr>
<td>Discard mortality rate</td>
<td>$\delta_d$</td>
<td>Proportion discards by fleet $d$ that die.</td>
</tr>
<tr>
<td>Observed discard mortalities</td>
<td>$\tilde{D}_{d,y}$</td>
<td>$D_{d,y} = \delta_d \tilde{D}_{d,y}$</td>
</tr>
<tr>
<td>CVs of dead discards</td>
<td>$CV_{D_{d,y}}^D$</td>
<td>Annual CV in arithmetic space.</td>
</tr>
<tr>
<td><strong>Population Model</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean length at age</td>
<td>$l_a$</td>
<td>Total length (midyear, in units of mm); $l_a = L_\infty(1 - \exp[-K(a - t_0 + \tau)])$ where $K$, $L_\infty$, and $t_0$ are parameters, and $\tau$ is a fixed value representing a fraction of the year (typically $\tau = 0.5$).</td>
</tr>
<tr>
<td>CV of $l_a$</td>
<td>$CV_{l_a}^\lambda$</td>
<td>Coefficient of variation of length at age.</td>
</tr>
<tr>
<td>SD of $l_a$</td>
<td>$\sigma_{l_a}^\lambda$</td>
<td>Standard deviation of length at age, $\sigma_{l_a}^\lambda = CV_{l_a}^\lambda l_a$.</td>
</tr>
<tr>
<td>Quantity</td>
<td>Symbol</td>
<td>Description or definition</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Age–length conversion of population</td>
<td>$\psi_{a,t}$</td>
<td>$\psi_{a,t=1} = \int_{x=0}^{\ell_{a,t=1}} f(x;l_0,\sigma_0^2)dx$, where $f$ represents the normal density function and $x$ is the variable of integration. The smallest length bin uses zero as the lower bound, and the largest uses infinity (i.e., is a plus group). Matrix $\psi$ is rescaled to sum to one within each age, which may be necessary because of truncating the normal distribution at a lower bound of zero.</td>
</tr>
<tr>
<td>Mean length at age of landings and discards</td>
<td>$\ell_{(f,d),a,y}$</td>
<td>BAM contains the option for fleet-specific mean lengths of landings ($\ell_{f,a,y}$) and discards ($\ell_{d,a,y}$), modeled with von Bertalanffy growth as for the population ($\ell_a$), but with separately estimated parameters. If this option is not used, $\ell_{(f,d),a,y} = \ell_a$.</td>
</tr>
<tr>
<td>Age–length conversion of landings</td>
<td>$\psi_{L,f,a,l,y}$</td>
<td>Computation is analogous to that of $\psi_{a,t}$, but with mean lengths based on $\ell_{f,a,y}$.</td>
</tr>
<tr>
<td>Age–length conversion of discards</td>
<td>$\psi_{D,d,a,l,y}$</td>
<td>Computation is analogous to that of $\psi_{a,t}$, but with mean lengths based on $\ell_{d,a,y}$. In addition, if a regulatory size limit ($l_{\text{lim}}$, $l_{\text{lim}}$) is in effect, the distribution of sizes at age may be truncated, such that $\psi_{D,d,a,l,y} &lt; l_{\text{lim}} = 0$. If so, annual matrices $\psi_{D,d,a,l,y}$ are rescaled to sum to one within each age.</td>
</tr>
<tr>
<td>Individual weight at age of population</td>
<td>$w_a$</td>
<td>Typically computed from mean length at age, for example using the equation $w_a = \theta_1 \ell_a^{\theta_2}$, where $\theta_1$ and $\theta_2$ are parameters.</td>
</tr>
<tr>
<td>Individual weight at age of landings and discards</td>
<td>$w_{(f,d),a,y}$</td>
<td>Computed from mean length at age, for example by $w_{(f,d),a,y} = \theta_1 \ell_{(f,d),a,y}^{\theta_2}$.</td>
</tr>
<tr>
<td>Natural mortality rate</td>
<td>$m_a$</td>
<td>Natural mortality rate at age, typically assumed constant across years.</td>
</tr>
<tr>
<td>Proportion female at age</td>
<td>$\eta_a$</td>
<td>A decreasing function for protogynous hermaphrodites; constant for gonochoristic stocks. Typically assumed constant across years.</td>
</tr>
<tr>
<td>Proportion male at age</td>
<td>$1 - \eta_a$</td>
<td>Complement of proportion female.</td>
</tr>
<tr>
<td>Proportion females mature at age</td>
<td>$m_a^\prime$</td>
<td>Typically increases with age.</td>
</tr>
<tr>
<td>Proportion males mature at age</td>
<td>$m'_a$</td>
<td>For protogynous hermaphrodites, all males typically assumed mature.</td>
</tr>
<tr>
<td>Batch fecundity at age</td>
<td>$E_a$</td>
<td>Eggs spawned per batch.</td>
</tr>
<tr>
<td>Number annual batches at age</td>
<td>$b_a$</td>
<td>Number of batches spawned per year.</td>
</tr>
<tr>
<td>Annual fecundity at age</td>
<td>$F_a$</td>
<td>$F_a = b_a E_a$. If fecundity information is unavailable, spawning biomass is used as a proxy.</td>
</tr>
<tr>
<td>Index date</td>
<td>$t_{\text{index}}$</td>
<td>Fraction denoting the proportional time of year for computing abundance at age, as used to predict indices of abundance. Typically, $t_{\text{index}} = 0.5$ for mid-year calculations.</td>
</tr>
<tr>
<td>Spawning date</td>
<td>$t_{\text{spawn}}$</td>
<td>Fraction denoting the proportional time of year when spawning occurs. For example, if peak spawning occurs at the end of March, $t_{\text{spawn}} = 0.25$.</td>
</tr>
</tbody>
</table>
| Reproductive capacity at age | $\xi_a$ | Multiplier on abundance to define reproductive output (spawning stock), for example: $\xi_a = \begin{cases} \eta_a m_a \lambda_F & : \text{If based on fecundity} \\
\eta_a m_a w_a & : \text{If based on mature female body weight} \\
\left[\eta_a m_a + (1 - \eta_a) m'_a\right] w_a & : \text{If based on mature body weight of both sexes.} \end{cases}$ |
<p>| Selectivities | $s_{(f,d),a,y}$ | Modeled as a free parameter at each age or using a parametric approach, as described in the “Selectivities” section of the text. |
| Fishing mortality rate of landings | $F_{f,a,y}$ | $F_{f,a,y} = s_{f,a,y} F_{f,y}$, where $F_{f,y}$ is the (estimated) fully selected fishing mortality rate of landings by fleet. |</p>
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Description or definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fishing mortality rate of discards</td>
<td>$F_{d,a,y}$</td>
<td>$F_{d,a,y} = s_{d,a,y} F_{d,y}$ where $F_{d,y}$ is the (estimated) fully selected fishing mortality rate of discards by fleet.</td>
</tr>
<tr>
<td>Total fishing mortality rate</td>
<td>$F_{a,y}$</td>
<td>$F_{a,y} = \sum_{(f,d)} F_{(f,d),a,y}$</td>
</tr>
<tr>
<td>Total mortality rate</td>
<td>$Z_{a,y}$</td>
<td>$Z_{a,y} = M_{a} + F_{a,y}$</td>
</tr>
<tr>
<td>Apical $F$</td>
<td>$F_{y}$</td>
<td>$F_{y} = \max_{a}(F_{a,y})$</td>
</tr>
<tr>
<td>Initialization mortality at age</td>
<td>$Z_{a,y}^{\text{init}}$</td>
<td>$Z_{a,y}^{\text{init}} = M_{a} + F_{a,y}^{\text{init}}$ where $F_{a,y}^{\text{init}}$ is the initialization $F$ at age.</td>
</tr>
<tr>
<td>Initial abundance at age per recruit at time of spawning</td>
<td>$N_{PRA}^{\text{init}}$</td>
<td>Same calculations as for $N_{PRA}$ (described below), but based on $Z_{a,y}^{\text{init}}$.</td>
</tr>
<tr>
<td>Initial spawners per recruit</td>
<td>$\phi_{\text{init}}$</td>
<td>$\phi_{\text{init}} = \sum_{a} N_{PRA}^{\text{init}} \xi_{a}$</td>
</tr>
<tr>
<td>Expected initial recruitment</td>
<td>$\bar{R}_{y1}$</td>
<td>$\bar{R}<em>{y1} = R^{eq}(\phi</em>{F} = \phi_{\text{init}})$ where $R^{eq}$ is as described in the text. If recruitment deviations are included in initial year $y1$, bias correction would be excluded from the calculation.</td>
</tr>
<tr>
<td>Initial equilibrium abundance at age</td>
<td>$N_{eq}^{a,y}$</td>
<td>Equilibrium age structure given $\bar{R}<em>{y1}$ and $Z</em>{a,y}^{\text{init}}$.</td>
</tr>
<tr>
<td>Abundance at age</td>
<td>$N_{a,y}$</td>
<td>$N_{a,y} = N_{eq}^{a,y} \forall a \in (2\ldots A)$</td>
</tr>
<tr>
<td>Abundance at age (partial year)</td>
<td>$N_{a,y}^{\prime}$</td>
<td>Used to match indices of abundance, $N_{a,y}^{\prime} = N_{a,y} \exp(-t_{\text{index}} Z_{a,y})$.</td>
</tr>
<tr>
<td>Abundance at age at time of spawning</td>
<td>$N_{a,y}^{\prime\prime}$</td>
<td>Assumed to correspond with peak spawning, $N_{a,y}^{\prime\prime} = \exp(-t_{\text{spawn}} Z_{a,y}) N_{a,y}$.</td>
</tr>
<tr>
<td>Unfished abundance at age per recruit at time of spawning</td>
<td>$N_{PRA}$</td>
<td>$N_{PRA} = 1 \times \exp(-t_{\text{spawn}} M_{1})$</td>
</tr>
<tr>
<td>Unfished spawners per recruit</td>
<td>$\phi_{0}$</td>
<td>$\phi_{0} = \sum_{a=1}^{A} N_{PRA} \xi_{a}$</td>
</tr>
<tr>
<td>Reproductive output</td>
<td>$S_{y}$</td>
<td>$\sum_{a=1}^{A} N_{a,y} \xi_{a}$, also called spawning stock.</td>
</tr>
<tr>
<td>Population biomass</td>
<td>$B_{y}$</td>
<td>$B_{y} = \sum_{a} N_{a,y} w_{a}$</td>
</tr>
<tr>
<td>Catchability</td>
<td>$q_{u,y}$</td>
<td>$q_{u,y} = q_{d}^{\text{density}}(B_{y}^{u}) f_{\text{trend}}^{u}(y) f_{\text{rw}}^{u}(y)$ The functions $f_{\text{density}}$, $f_{\text{trend}}^{u}$, and $f_{\text{rw}}^{u}$ are not required but allow for effects of density, time (e.g., technology creep), and random walk, as applied when fitting indices of abundance (see section “Catchability”).</td>
</tr>
<tr>
<td>Landing at age in numbers</td>
<td>$l_{f,a,y}^{u}$</td>
<td>$l_{f,a,y}^{u} = \frac{F_{f,a,y}}{Z_{a,y}} N_{a,y} \left[1 - \exp(-Z_{a,y})\right]$</td>
</tr>
</tbody>
</table>
Table 1. (continued)

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Description or definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landing at age in weight</td>
<td>$l_{f,a,y}'$</td>
<td>$l_{f,a,y}' = Cw_{f,a,y}l_{f,a,y}'$, where the constant $C$ converts from units of $w$ to those of observed removals.</td>
</tr>
<tr>
<td>Discard mortalities at age in numbers</td>
<td>$d_{d,a,y}'$</td>
<td>$d_{d,a,y}' = F_{d,a,y}N_{a,y}[1 - \exp(-Z_{a,y})]$</td>
</tr>
<tr>
<td>Discard mortalities at age in weight</td>
<td>$d_{d,a,y}''$</td>
<td>$d_{d,a,y}'' = Cw_{d,a,y}d_{d,a,y}'$, where the constant $C$ converts from units of $w$ to those of observed removals.</td>
</tr>
<tr>
<td>Predicted total landings in numbers</td>
<td>$L_{f,y}'$</td>
<td>$L_{f,y}' = \sum_{a} l_{f,a,y}'$</td>
</tr>
<tr>
<td>Predicted total landings in weight</td>
<td>$L_{f,y}''$</td>
<td>$L_{f,y}'' = \sum_{a} l_{f,a,y}''$</td>
</tr>
<tr>
<td>Predicted discard mortalities in numbers</td>
<td>$D_{d,y}'$</td>
<td>$D_{d,y}' = \sum_{a} d_{d,a,y}'$</td>
</tr>
<tr>
<td>Predicted discard mortalities in weight</td>
<td>$D_{d,y}''$</td>
<td>$D_{d,y}'' = \sum_{a} d_{d,a,y}''$</td>
</tr>
<tr>
<td>Predicted length compositions of fishery independent data</td>
<td>$p_{u,a,y}^\lambda$</td>
<td>$p_{u,a,y}^\lambda = \frac{\sum_{a} \psi_{u,a,y} s_{u,a,y} N_{a,y}'}{\sum_{a} s_{u,a,y} N_{a,y}'}$</td>
</tr>
<tr>
<td>Predicted length compositions of landings</td>
<td>$p_{f,a,y}^\lambda$</td>
<td>$p_{f,a,y}^\lambda = \frac{\sum_{a} \psi_{f,a,y} s_{u,a,y} N_{a,y}'}{\sum_{a} s_{u,a,y} N_{a,y}'}$</td>
</tr>
<tr>
<td>Predicted length compositions of discards</td>
<td>$p_{d,a,y}^\lambda$</td>
<td>$p_{d,a,y}^\lambda = \frac{\sum_{a} \psi_{d,a,y} s_{u,a,y} N_{a,y}'}{\sum_{a} s_{u,a,y} N_{a,y}'}$</td>
</tr>
<tr>
<td>Predicted age compositions of indices</td>
<td>$p_{a,a,y}^\alpha$</td>
<td>$p_{a,a,y}^\alpha = \frac{\sum_{a} \psi_{a,a,y}}{\sum_{a} s_{a,a,y} N_{a,y}'}$</td>
</tr>
<tr>
<td>Predicted age compositions of landings</td>
<td>$p_{f,a,y}^\alpha$</td>
<td>$p_{f,a,y}^\alpha = \frac{\sum_{a} \psi_{f,a,y}}{L_{f,a,y}'}$</td>
</tr>
<tr>
<td>Predicted CPUE</td>
<td>$U_{u,y}$</td>
<td>$U_{u,y} = \begin{cases} q_{u,y} \sum_{a} w_{u,a,y} s_{u,a,y} N_{a,y}' &amp; : \text{if in weight} \ q_{u,y} \sum_{a} s_{u,a,y} N_{a,y}' &amp; : \text{if in numbers} \end{cases}$</td>
</tr>
</tbody>
</table>

where $s_{u,a,y}$ is the selectivity of the relevant fleet or survey, and $w_{u,a,y}$ is the relevant weight at age (either $w$ or $w_{f,a,y}'$).

Objective Function
(Equations in §3)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Landings</td>
<td>$\Lambda^L$</td>
<td>Lognormal likelihood.</td>
</tr>
<tr>
<td>Discards</td>
<td>$\Lambda^D$</td>
<td>Lognormal likelihood.</td>
</tr>
<tr>
<td>Indices of abundance</td>
<td>$\Lambda^U$</td>
<td>Lognormal likelihood.</td>
</tr>
<tr>
<td>Age compositions</td>
<td>$\Lambda^\alpha$</td>
<td>Robust multinomial likelihood.</td>
</tr>
<tr>
<td>Length compositions</td>
<td>$\Lambda^\lambda$</td>
<td>Robust multinomial likelihood.</td>
</tr>
<tr>
<td>Recruitment deviations</td>
<td>$\Lambda^R_1$</td>
<td>Lognormal likelihood.</td>
</tr>
<tr>
<td>Early recruitment deviations</td>
<td>$\Lambda^R_2$</td>
<td>Optional sum of squares penalty for deviations early in the time series.</td>
</tr>
<tr>
<td>Quantity</td>
<td>Symbol</td>
<td>Description or definition</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Late recruitment deviations</td>
<td>$\Lambda^{R3}$</td>
<td>Optional sum of squares penalty for deviations late in the time series.</td>
</tr>
<tr>
<td>Initial age structure devia-</td>
<td>$\Lambda^{init}$</td>
<td>Optional sum of squares penalty.</td>
</tr>
<tr>
<td>tions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catchability random walk</td>
<td>$\Lambda^{q}$</td>
<td>Optional sum of squares penalty.</td>
</tr>
<tr>
<td>Fishing rate</td>
<td>$\Lambda^{F}$</td>
<td>Optional penalty on very large values of apical F.</td>
</tr>
<tr>
<td>Miscellaneous parameters</td>
<td>$\Lambda^{P}$</td>
<td>Sum of optional penalties (a.k.a., priors) applied to any parameter. Each penalty may take the form of a normal, lognormal, or beta likelihood.</td>
</tr>
<tr>
<td>Total objective function</td>
<td>$\Lambda$</td>
<td>$\Lambda = \Lambda^{L} + \Lambda^{D} + \Lambda^{U} + \Lambda^{a} + \Lambda^{h} + \Lambda^{R1} + \Lambda^{R2} + \Lambda^{R3} + \Lambda^{init} + \Lambda^{q} + \Lambda^{F} + \Lambda^{P}$, the overall objective function minimized by the assessment model. Likelihood components are negative likelihoods or negative log-likelihoods.</td>
</tr>
</tbody>
</table>
Figure 1. Flow of operations in the Beaufort Assessment Model.

Start → Input data → Initialize parameter values → Update parameter values

Growth → Reproductive capacity → Length, weight at age → spr(F=0) → Selectivities

Objective function → Mortality rates

NO

Age, length compositions → Catchability, indices → Landings, discards → Abundance at age → Recruits, bias correction

Current average selectivities → Biological reference points → Output results (R format) → End

Convergence? → YES
9 AD Model Builder code to implement the Beaufort Assessment Model. Example application toward gag.

```plaintext
//##--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><-->
//## Gag Grouper assessment January 2014
//## 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 nyr;
number nyr_rec;
//this section MUST BE INDENTED!!!
LOCAL_CALCS
  nyr=endyr-styr+1.;
  nyr_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 biasCor;

//##--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><-->
//-- BAM DATA_SECTION: observed data section
//--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><-->
//################Commercial handline fleet #=======================================================================
// Commercial CFUE
init_int styr_CH_cpfue;
init_int endyr_CH_cpfue;
init_vector obs_CH_cpfue(styr_CH_cpfue,endyr_CH_cpfue); //Observed CPUE
init_vector cH_cpfue_cv(styr_CH_cpfue,endyr_CH_cpfue); //CV of cpue

// Commercial Landings (1000 lb gutted weight)
init_int styr_CH_L;
init_int endyr_CH_L;
init_vector obs_CH_L(cpfue(styr_CH_L,endyr_CH_L),cpfue(styr_CH_L,endyr_CH_L)); //CV of cpue

// Commercial Discards (1000 fish)
init_int styr_CH_D;
init_int endyr_CH_D;
init_vector obs_CH_released(cpfue(styr_CH_D,endyr_CH_D),cpfue(styr_CH_D,endyr_CH_D));

// Commercial Length Compositions (3 cm bins)
init_int nyr_CH_len;
```
init_vector yrr_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 HL 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 diving fleet #######################################
// Comm DV Landings (1000 lb gutted weight)
init_int styr_cD_L;
init_int endyr_cD_L;
init_vector obs_cD_L(styr_cD_L,endyr_cD_L);
init_vector cD_L_cv(styr_cD_L,endyr_cD_L);

// Comm DV length Compositions (3 cm bins)
init_int nyr_cD_lenc;
init_ivector yrs_cD_lenc(1,nyr_cD_lenc);
init_vector nsamp_cD_lenc(1,nyr_cD_lenc);
init_vector nfish_cD_lenc(1,nyr_cD_lenc);
init_matrix obs_cD_lenc(1,nyr_cD_lenc,1,nlenbins);

// Comm DV age compositions
init_int nyr_cD_agec;
init_ivector yrs_cD_agec(1,nyr_cD_agec);
init_vector nsamp_cD_agec(1,nyr_cD_agec);
init_vector nfish_cD_agec(1,nyr_cD_agec);
init_matrix obs_cD_agec(1,nyr_cD_agec,1,nages_agec);

//###################Headboat fleet ##########################################
// HB CPUE
init_int styr_HB_cpue;
init_int endyr_HB_cpue;
init_vector obs_HB_cpue(styr_HB_cpue,endyr_HB_cpue);
init_vector HB_cpue_cv(styr_HB_cpue,endyr_HB_cpue);

// HB Landings (1000s fish)
init_int styr_HB_L;
init_int endyr_HB_L;
init_vector obs_HB_L(styr_HB_L,endyr_HB_L);
init_vector HB_L_cv(styr_HB_L,endyr_HB_L);

// HB Discards (1000 fish)
init_int styr_HB_D;
init_int endyr_HB_D;
init_vector obs_HB_released(styr_HB_D,endyr_HB_D);
init_vector HB_D_cv(styr_HB_D,endyr_HB_D);

// HB length Compositions (3 cm bins)
init_int nyr_HB_lenc;
init_ivector yrs_HB_lenc(1,nyr_HB_lenc);
init_vector nsamp_HB_lenc(1,nyr_HB_lenc);
init_vector nfish_HB_lenc(1,nyr_HB_lenc);
init_matrix obs_HB_lenc(1,nyr_HB_lenc,1,nlenbins);

// HB age compositions
init_int nyr_HB_agec;
init_ivector yrs_HB_agec(1,nyr_HB_agec);
init_vector nsamp_HB_agec(1,nyr_HB_agec);
init_vector nfish_HB_agec(1,nyr_HB_agec);
init_matrix obs_HB_agec(1,nyr_HB_agec,1,nages_agec);

//###################General Recreational fleet ##########################################
// MRFSS CPUE
init_int styr_GR_cpue;
init_int endyr_GR_cpue;
init_vector obs_GR_cpue(styr_GR_cpue,endyr_GR_cpue);
init_vector GR_cpue_cv(styr_GR_cpue,endyr_GR_cpue);

// GR Landings (1000s fish)
init_int styr_GR_L;
init_int endyr_GR_L;
init_vector obs_GR_L(styr_GR_L,endyr_GR_L);
init_vector GR_L_cv(styr_GR_L,endyr_GR_L);

// GR Discards (1000 fish)
init_int styr_GR_D;
init_int endyr_GR_D;
init_vector obs_GR_released(styr_GR_D,endyr_GR_D);
init_vector GR_D_cv(styr_GR_D,endyr_GR_D);

// GR length Compositions (3 cm bins)
init_int nyr_GR_lenc;
init_ivector yrs_GR_lenc(1,nyr_GR_lenc);
init_vector nsamp_GR_lenc(1,nyr_GR_lenc);
init_vector nfish_GR_lenc(1,nyr_GR_lenc);
init_matrix obs_GR_lenc(1,nyr_GR_lenc,1,nlenbins);

// GR age compositions
init_int nyr_GR_agec;
init_ivector yrs_GR_agec(1,nyr_GR_agec);
init_vector nsamp_GR_agec(1,nyr_GR_agec);
init_vector nfish_GR_agec(1,nyr_GR_agec);
init_matrix obs_GR_agec(1,nyr_GR_agec,1,nages_agec);

---

// 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);
init_vector set_len_cv(1,7); //CV of length at age and its standard error all fish
init_vector set_M_constant(1,7); //Scalar used only for computing MSST.

//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_cH1(1,7);
init_vector set_selpar_slope_cH1(1,7);
init_vector set_selpar_L50_cH2(1,7);
init_vector set_selpar_slope_cH2(1,7);
init_vector set_selpar_L50_cH3(1,7);
init_vector set_selpar_slope_cH3(1,7);
init_vector set_selpar_L50_cD(1,7);
init_vector set_selpar_slope_cD(1,7);
init_vector set_selpar_afull_cD(1,7);
init_vector set_selpar_sigma_cD(1,7);
init_vector set_selpar_L50_HB1(1,7);
init_vector set_selpar_slope_HB1(1,7);
init_vector set_selpar_L50_HB2(1,7);
init_vector set_selpar_slope_HB2(1,7);
init_vector set_selpar_L50_HB3(1,7);
init_vector set_selpar_slope_HB3(1,7);

//--index catchability-----------------------------------------------------------------------------------
init_vector set_log_q_cH(1,7); //catchability coefficient (log) for comm handline index
init_vector set_log_q_HB(1,7); //catchability coefficient (log) for headboat index
init_vector set_log_q_GR(1,7); //catchability coefficient (log) for general rec 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_cD(1,7);
init_vector set_log_avg_F_HB(1,7);
init_vector set_log_avg_F_GR(1,7);
init_vector set_log_avg_F_cH_D(1,7);
init_vector set_log_avg_F_HB_D(1,7);
init_vector set_log_avg_F_GR_D(1,7);

//##################Dev Vector Parameter values (vals) and bounds #################################
//--F vectors---------------------------
init_vector set_log_F_dev_cH(1,3);
init_vector set_log_F_dev_cD(1,3);
init_vector set_log_F_dev_HB(1,3);
init_vector set_log_F_dev_GR(1,3);
init_vector set_log_F_dev_cH_D(1,3);
init_vector set_log_F_dev_HB_D(1,3);
init_vector set_log_F_dev_GR_D(1,3);
init_vector set_log_rec_dev(1,3);
init_vector set_log_Nage_dev(1,3);

//--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><)--><--}
init_number huge_number; //huge number, to avoid irregular parameter space
init_number dzero; //small additive constant to prevent division by zero
init_number mt2lb; //conversion of metric tons to lb
init_number mt2klb; //conversion of metric tons to 1000 lb
init_number g2klb; //conversion of grams to 1000 lb
init_number g2kg; //conversion of grams to kg
init_number sqrt2pi;
init_int ff;
init_int iyear;
// #######Indexing integers for year(iyear), age(iage),length(ilen) ###############
init_matrix age_error(1,nages,1,nages);
//ageing error matrix (columns are true ages, rows are ages as read for age comps: columns must sum to one)
init_number minSS_HB_agec;
init_number minSS_cD_agec;
//threshold sample sizes for inclusion of age comps
init_number minSS_HB_lenc;
init_number minSS_cD_lenc;
//threshold sample sizes for inclusion of length comps
init_number minSS_cH_agec;
init_number set_Ftune;
//Tune Fapex (if applied, tuning removed in final phase of optimization)
init_number set_q_RW_rec_var; //variance of RW q
init_int set_q_RW_phase; //value sets estimation phase of random walk on fishery q's
init_int set_q_DD_phase; //value sets estimation phase of rate increase, negative value turns it off
//rate of increase on q
init_int set_q_rate_phase; //value sets estimation phase of rate increase, negative value turns it off
init_int SR_switch;
//Spawner-recruit parameters (Initial guesses or fixed values)
init_number set_Dmort_GR;
init_number set_Dmort_HB;
init_number set_Dmort_cH;
//discard mortality constants
init_number max_obs_age; //max observed age, used to scale M, if estimated
init_vector set_M(1,nages); //age-dependent: used in model
init_number spawn_time_frac; //time of year of peak spawning, as a fraction of the year
const double selpar_slope_HB3_LO=set_selpar_slope_HB3(2); const double selpar_slope_HB3_HI=set_selpar_slope_HB3(3); const double selpar_slope_HB3_PH=set_selpar_slope_HB3(4);
const double selpar_L50_HB3_LO=set_selpar_L50_HB3(2); const double selpar_L50_HB3_HI=set_selpar_L50_HB3(3); const double selpar_L50_HB3_PH=set_selpar_L50_HB3(4);
const double selpar_slope_HB2_LO=set_selpar_slope_HB2(2); const double selpar_slope_HB2_HI=set_selpar_slope_HB2(3); const double selpar_slope_HB2_PH=set_selpar_slope_HB2(4);
const double selpar_L50_HB2_LO=set_selpar_L50_HB2(2); const double selpar_L50_HB2_HI=set_selpar_L50_HB2(3); const double selpar_L50_HB2_PH=set_selpar_L50_HB2(4);
const double selpar_slope_HB1_LO=set_selpar_slope_HB1(2); const double selpar_slope_HB1_HI=set_selpar_slope_HB1(3); const double selpar_slope_HB1_PH=set_selpar_slope_HB1(4);
const double selpar_L50_HB1_LO=set_selpar_L50_HB1(2); const double selpar_L50_HB1_HI=set_selpar_L50_HB1(3); const double selpar_L50_HB1_PH=set_selpar_L50_HB1(4);
const double selpar_slope_cD_LO=set_selpar_slope_cD(2); const double selpar_slope_cD_HI=set_selpar_slope_cD(3); const double selpar_slope_cD_PH=set_selpar_slope_cD(4);
const double selpar_L50_cD_LO=set_selpar_L50_cD(2); const double selpar_L50_cD_HI=set_selpar_L50_cD(3); const double selpar_L50_cD_PH=set_selpar_L50_cD(4);
const double selpar_slope_cH3_LO=set_selpar_slope_cH3(2); const double selpar_slope_cH3_HI=set_selpar_slope_cH3(3); const double selpar_slope_cH3_PH=set_selpar_slope_cH3(4);
const double selpar_L50_cH3_LO=set_selpar_L50_cH3(2); const double selpar_L50_cH3_HI=set_selpar_L50_cH3(3); const double selpar_L50_cH3_PH=set_selpar_L50_cH3(4);
const double selpar_slope_cH2_LO=set_selpar_slope_cH2(2); const double selpar_slope_cH2_HI=set_selpar_slope_cH2(3); const double selpar_slope_cH2_PH=set_selpar_slope_cH2(4);
const double selpar_slope_cH1_LO=set_selpar_slope_cH1(2); const double selpar_slope_cH1_HI=set_selpar_slope_cH1(3); const double selpar_slope_cH1_PH=set_selpar_slope_cH1(4);
const double selpar_L50_cH1_LO=set_selpar_L50_cH1(2); const double selpar_L50_cH1_HI=set_selpar_L50_cH1(3); const double selpar_L50_cH1_PH=set_selpar_L50_cH1(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 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 steep_LO=set_steep(2); const double steep_HI=set_steep(3); const double steep_PH=set_steep(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 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 K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
const double log_q_GR_LO=set_log_q_GR(2); const double log_q_GR_HI=set_log_q_GR(3); const double log_q_GR_PH=set_log_q_GR(4);
const double log_q_HB_LO=set_log_q_HB(2); const double log_q_HB_HI=set_log_q_HB(3); const double log_q_HB_PH=set_log_q_HB(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_q_GR_LO=set_log_q_GR(2); const double log_q_GR_HI=set_log_q_GR(3); const double log_q_GR_PH=set_log_q_GR(4);
const double log_q_HB_LO=set_log_q_HB(2); const double log_q_HB_HI=set_log_q_HB(3); const double log_q_HB_PH=set_log_q_HB(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);

22
vector nfish_cD_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
matrix nfish_cH_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)

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

vector nfish_cD_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
vector nfish_cH_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)

///////////////////////////////Recreational (HB and GR)-----------------------------------------

vector nfish_cD_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
vector nfish_cH_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)

/////////////////////////////////////////////////////Commercial diving--------------------------------

vector nfish_cD_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
vector nfish_cH_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)

//Computed effective sample size for output (not used in fitting)

vector nfish_cD_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
vector nfish_cH_agec_allyr(styr,endyr); //Used to compute spawning biomass (total mature biomass: males + females)
Discards (number dead fish)

Landings in numbers (total or 1000 fish) and in wgt (1000 lb gutted)

Fishery dependent catchability over time, may be constant

Fishery dependent random walk catchability

Catchability (CPUE q's)

Weighted total selectivity (effort-weighted, recent selectivities)

Landings in numbers (total or 1000 fish) and in wgt (1000 lb gutted)

Discards (number dead fish)
vector log_F_dev_HB_out(styr_HB_L,endyr_HB_L);

init_bounded_dev_vector log_F_dev_HB(styr_HB_L,endyr_HB_L,log_F_dev_HB_LO,log_F_dev_HB_HI,log_F_dev_HB_PH);

init_bounded_number log_avg_F_HB(log_avg_F_HB_LO,log_avg_F_HB_HI,log_avg_F_HB_PH);

number log_F_dev_end_cD;

vector F_cD_out(styr,endyr); //used for intermediate calculations in fcn get_mortality

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

vector log_F_dev_cD_out(styr_cD_L,endyr_cD_L);

init_bounded_number log_avg_F_cD(log_avg_F_cD_LO,log_avg_F_cD_HI,log_avg_F_cD_PH);

number log_F_dev_end_cH;

number log_F_dev_init_cH;

vector F_cH_out(styr,endyr); //used for intermediate calculations in fcn get_mortality

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

vector log_F_dev_cH_out(styr_cH_L,endyr_cH_L);

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_avg_F_cH_out(1,8);

init_bounded_number log_avg_F_cH(log_avg_F_cH_LO,log_avg_F_cH_HI,log_avg_F_cH_PH);

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

vector Fapex(styr,endyr); //Max across ages, fishing mortality rate by year (may differ from Fsum bc of dome-shaped selex

vector Fsum(styr,endyr); //Full fishing mortality rate by year

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

number smsy2msst75; //scales Smsy to get msst using 75%. Used only in output.

number smsy2msst; //scales Smsy to get msst using (1-M). Used only in output.

vector M_constant_out(1,8);

init_bounded_number M_constant(M_constant_LO,M_constant_HI,M_constant_PH); //age-indpendent: used only for MSST

vector M(1,nages); //age-dependent natural mortality

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

number wgt_wgted_D_denom; //used in intermediate calculations

number wgt_wgted_L_denom; //used in intermediate calculations

vector wgt_wgted_D_klb(1,nages); //fishery-weighted average weight at age of discards in gutted weight

vector wgt_wgted_L_klb(1,nages); //fishery-weighted average weight at age of landings in gutted weight

number FdF_msy_end_mean; //geometric mean of last X yrs

number FdF_msy_end;

number SdSSB_msy_end;

vector SdSSB_msy(styr,endyr+1);

vector FdF_msy(styr,endyr);

vector SSB_eq(1,n_iter_msy); //equilibrium reproductive capacity values corresponding to F values in F_msy

vector D_eq_knum(1,n_iter_msy); //equilibrium discards(1000 fish) values corresponding to F values in F_msy

vector D_eq_klb(1,n_iter_msy); //equilibrium discards(klb gutted wgt) 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 L_eq_klb(1,n_iter_msy); //equilibrium landings(klb gutted wgt) 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 F_D_age_msy(1,nages); //fishing mortality of discards at age for MSY calculations

vector F_L_age_msy(1,nages); //fishing mortality landings (not discards) at age for MSY calculations

vector Z_age_msy(1,nages); //total mortality at age for MSY calculations

vector D_age_msy(1,nages); //discard mortality (dead discards) at age for MSY calculations

vector L_age_msy(1,nages); //landings at age for MSY calculations

number spr_msy_out; //spr at F=Fmsy

number B_msy_out; //total biomass at MSY

number D_msy_knum_out; //discards associated with msy (1000 fish)

number D_msy_klb_out; //discards associated with msy (1000 lb gutted weight)

number msy_knum_out; //max sustainable yield (1000 fish)

number msy_klb_out; //max sustainable yield (1000 lb gutted weight)

number SSB_msy_out; //SSB (total mature biomass) at msy

number F_end_D(1,nages);

number F_end_L(1,nages);

number F_end(1,nages);

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

vector F_D(styr,endyr); //Discards at age

vector F_L(styr,endyr); //Landings at age

vector F(styr,endyr); //Totals at age
matrix F_HB(styr,endyr,1,nages);  //used for intermediate calculations in fmcn get_mortality
number log_F_dev_init_HB;
number log_F_dev_end_HB;
init_bounded_number log_avg_F_GR(log_avg_F_GR_LO,log_avg_F_GR_HI,log_avg_F_GR_PH);
vector log_avg_F_GR_out(1,8);
init_bounded_dev_vector log_F_dev_GR(styr_GR_L,endyr_GR_L,log_F_dev_GR_LO,log_F_dev_GR_HI,log_F_dev_GR_PH);
vector log_F_dev_GR_out(styr_GR_L,endyr_GR_L);
matrix F_GR(styr,endyr,1,nages);
vector F_GR_out(styr,endyr);  //used for intermediate calculations in fcn get_mortality
number log_F_dev_init_GR;
number log_F_dev_end_GR;
init_bounded_number log_avg_F_cH_D(log_avg_F_cH_D_LO,log_avg_F_cH_D_HI,log_avg_F_cH_D_PH);
vector log_avg_F_cH_D_out(1,8);
init_bounded_dev_vector log_F_dev_cH_D(styr_cH_D,endyr_cH_D,log_F_dev_cH_D_LO,log_F_dev_cH_D_HI,log_F_dev_cH_D_PH);
vector log_F_dev_cH_D_out(styr_cH_D,endyr_cH_D);
matrix F_cH_D(styr,endyr,1,nages);
vector F_cH_D_out(styr,endyr);  //used for intermediate calculations in fcn get_mortality
number log_F_dev_end_cH_D;
init_bounded_number log_avg_F_HB_D(log_avg_F_HB_D_LO,log_avg_F_HB_D_HI,log_avg_F_HB_D_PH);
vector log_avg_F_HB_D_out(1,8);
init_bounded_dev_vector log_F_dev_HB_D(styr_HB_D,endyr_HB_D,log_F_dev_HB_D_LO,log_F_dev_HB_D_HI,log_F_dev_HB_D_PH);
vector log_F_dev_HB_D_out(styr_HB_D,endyr_HB_D);
matrix F_HB_D(styr,endyr,1,nages);
vector F_HB_D_out(styr,endyr);  //used for intermediate calculations in fcn get_mortality
number log_F_dev_end_HB_D;
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);
number F_init_denom;  //interim calculation

////---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 gutted)-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_cH;
number sdnr_lc_cD;
number sdnr_ac_cH;
number sdnr_ac_cD;
number sdnr_I_cH;
number sdnr_I_HB;
number sdnr_I_GR;

////-------Objective function components------------------------------------------------------------------
number w_L;
number w_D;
number w_I_cH;
number w_I_HB;
number w_I_GR;
number w_lc_cH;
number w_lc_cD;
number w_lc_HB;
number w_ac_cH;
number w_ac_cD;
number w_ac_HB;
number w_Nage_init;
number w_rec;
number w_rec_early;
number w_rec_end;
number w_fullF;
number w_Ftune;
number f_cH_L;
number f_cD_L;
number f_HB_L;
number v_cH_L;
number v_cD_L;
number v_HB_L;
number v_L;
number v_D;
number v_I_cH;
number v_I_HB;
number v_I_GR;
number v_lc_cH;
number v_lc_cD;
number v_lc_HB;
number v_ac_cH;
number v_ac_cD;
number v_ac_HB;
number v_Nage_init;
number v_w_rec;
number v_w_rec_early;
number v_w_rec_end;
number v_w_fullF;
number v_w_Ftune;
number v_f_cH_L;
number v_f_cD_L;
number v_f_HB_L;

26
number f_cH_D;  
number f_HB_D;  
number f_GR_D;  
number f_cH_cpue;  
number f_HB_cpue;  
number f_GR_cpue;  
number f_cH_lenc;  
number f_cD_lenc;  
number f_HB_lenc;  
number f_cH_agec;  
number f_cD_agec;  
number f_HB_agec;  

// Penalties and constraints. Not all are used.
number f_fape = 0;  //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_priors;  //prior information on parameters
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 R-compatible output functions. Comment line if not using admb2r.
#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
Dmort_cH=set_Dmort_cH;
Dmort_HB=set_Dmort_HB;
Dmort_GR=set_Dmort_GR;

//Dead discards
obs_cH_D=Dmort_cH*obs_cH_released;
obs_HB_D=Dmort_HB*obs_HB_released;
obs_GR_D=Dmort_GR*obs_GR_released;

//Growth parameters
Linf=set_Linf(1);
K=set_K(1);
t0=set_t0(1);
len_cv_val=set_len_cv(1);

//Maturity and sex transition
maturity_f=maturity_f_obs;
maturity_m=maturity_m_obs;
prop_f=prop_f_obs;

//Natural mortality
M=set_M;
M_constant=set_M_constant(1);
msy2msst=1.0-M_constant;
msy2msst75=0.75;

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

//Catchability
log_q_cH=set_log_q_cH(1);
log_q_HB=set_log_q_HB(1);
log_q_GR=set_log_q_GR(1);

q_rate=set_q_rate;
q_rate_fcn_cH=1.0;
q_rate_fcn_HB=1.0;
q_rate_fcn_GR=1.0;
q_DD_beta=set_q_DD_beta;
q_DD_fcn=1.0;
q_RW_log_dev_cH.initialize();
q_RW_log_dev_HB.initialize();
q_RW_log_dev_GR.initialize();

if (set_q_rate_phase<0 & q_rate!=0.0)
{
    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);}
    }
    for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
    {
        if (iyear>styr_HB_cpue & iyear <=2003)
        {
            //q_rate_fcn_HB(iyear)=(1.0+q_rate)*q_rate_fcn_HB(iyear-1); //compound
            q_rate_fcn_HB(iyear)=(1.0+(iyear-styr_HB_cpue)*q_rate)*q_rate_fcn_HB(styr_HB_cpue); //linear
        }
        if (iyear>2003) {q_rate_fcn_HB(iyear)=q_rate_fcn_HB(iyear-1);}
    }
    for (iyear=styr_GR_cpue; iyear<=endyr_GR_cpue; iyear++)
    {
        if (iyear>styr_GR_cpue & iyear <=2003)
        {
            //q_rate_fcn_GR(iyear)=(1.0+q_rate)*q_rate_fcn_GR(iyear-1); //compound
            q_rate_fcn_GR(iyear)=(1.0+(iyear-styr_GR_cpue)*q_rate)*q_rate_fcn_GR(styr_GR_cpue); //linear
        }
        if (iyear>2003) {q_rate_fcn_GR(iyear)=q_rate_fcn_GR(iyear-1);}
    }
} //end q_rate conditional

//Objective function weights
w_L=set_w_L;
w_D=set_w_D;
w_I_cH=set_w_I_cH;
w_I_HB=set_w_I_HB;
w_I_GR=set_w_I_GR;
w_lc_cH=set_w_lc_cH;
w_lc_cD=set_w_lc_cD;
w_lc_HB=set_w_lc_HB;
w_ac_cH=set_w_ac_cH;
w_ac_cD=set_w_ac_cD;
w_ac_HB=set_w_ac_HB;
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;
Ftune=set_w_Ftune;

//Fishing rates
F_init=set_F_init(1);
log_avg_F_cH=set_log_avg_F_cH(1);
log_avg_F_cD=set_log_avg_F_cD(1);
log_avg_F_HB=set_log_avg_F_HB(1);
log_avg_F_GR=set_log_avg_F_GR(1);
log_avg_F_cH_D=set_log_avg_F_cH_D(1);
log_avg_F_HB_D=set_log_avg_F_HB_D(1);
log_avg_F_GR_D=set_log_avg_F_GR_D(1);

//Selectivity parameters
selpar_L50_cH1=set_selpar_L50_cH1(1);
selpar_slope_cH1=set_selpar_slope_cH1(1);
selpar_L50_cH2=set_selpar_L50_cH2(1);
selpar_slope_cH2=set_selpar_slope_cH2(1);
selpar_L50_cH3=set_selpar_L50_cH3(1);
selpar_slope_cH3=set_selpar_slope_cH3(1);
selpar_L50_cD=set_selpar_L50_cD(1);
selpar_slope_cD=set_selpar_slope_cD(1);
selpar_afull_cD=set_selpar_afull_cD(1);
selpar_sigma_cD=set_selpar_sigma_cD(1);
selpar_L50_HB1=set_selpar_L50_HB1(1);
selpar_slope_HB1=set_selpar_slope_HB1(1);
selpar_L50_HB2=set_selpar_L50_HB2(1);
selpar_slope_HB2=set_selpar_slope_HB2(1);
selpar_L50_HB3=set_selpar_L50_HB3(1);
selpar_slope_HB3=set_selpar_slope_HB3(1);

//Conversions and fixed quantities
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;
iter_inc_msy=max_F_spr_msy/(n_iter_msy-1);
iter_inc_spr=max_F_spr_msy/(n_iter_spr-1);
// Fill in sample sizes of comps, possibly sampled in nonconsec yrs
// Used primarily for output in R object

nsamp_cH_lenc_allyr=missing;
nsamp_cD_lenc_allyr=missing;
nsamp_HB_lenc_allyr=missing;
nsamp_cH_agec_allyr=missing;
nsamp_cD_agec_allyr=missing;
nsamp_HB_agec_allyr=missing;
nfish_cH_lenc_allyr=missing;
nfish_cD_lenc_allyr=missing;
nfish_HB_lenc_allyr=missing;
nfish_cH_agec_allyr=missing;
nfish_cD_agec_allyr=missing;
nfish_HB_agec_allyr=missing;

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_cD_lenc; iyear++)
{if (nsamp_cD_lenc(iyear)>=minSS_cD_lenc)
 {nsamp_cD_lenc_allyr(yrs_cD_lenc(iyear))=nsamp_cD_lenc(iyear);
nfish_cD_lenc_allyr(yrs_cD_lenc(iyear))=nfish_cD_lenc(iyear);}}

for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
{if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
 {nsamp_HB_lenc_allyr(yrs_HB_lenc(iyear))=nsamp_HB_lenc(iyear);
nfish_HB_lenc_allyr(yrs_HB_lenc(iyear))=nfish_HB_lenc(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_cD_agec; iyear++)
{if (nsamp_cD_agec(iyear)>=minSS_cD_agec)
 {nsamp_cD_agec_allyr(yrs_cD_agec(iyear))=nsamp_cD_agec(iyear);
nfish_cD_agec_allyr(yrs_cD_agec(iyear))=nfish_cD_agec(iyear);}}

for (iyear=1; iyear<=nyr_HB_agec; iyear++)
{if (nsamp_HB_agec(iyear)>=minSS_HB_agec)
 {nsamp_HB_agec_allyr(yrs_HB_agec(iyear))=nsamp_HB_agec(iyear);
nfish_HB_agec_allyr(yrs_HB_agec(iyear))=nfish_HB_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_cH.initialize(); L_cH_num.initialize();
F_cD.initialize(); L_cD_num.initialize();
F_HB.initialize(); L_HB_num.initialize();
F_GR.initialize(); L_GR_num.initialize();
F_cH_D.initialize(); D_cH_num.initialize();
F_HB_D.initialize(); D_HB_num.initialize();
F_GR_D.initialize(); D_GR_num.initialize();
F_cH_out.initialize();
F_cD_out.initialize();
F_HB_out.initialize();
F_GR_out.initialize();

sel_cH.initialize();
sel_cD.initialize();
sel_HB.initialize();
F_cH_D.initialize();

log_rec_dev_output.initialize();
log_rec_devset=log_rec_devvals;
log_Hage_dev_output.initialize();
log_Hage_devset=log_Hage_devvals;

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

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

gt_length_weight_at_age();
gt_reprod();
gt_length_at_age_dist();
gt_weight_at_age_landings();
gt_spr_F0();

29
get_selectivity();
get_mortality();
get_bio_corr();
get_numbers_at_age();
get_landings_numbers();
get_landings_vgt();
get_deal_discard();
get_catchability_fncs();
get_indexs();
get_length_comps();
get_age_comps();
evaluate_objective_function();

FUNCTION get_length_weight_at_age
//compute mean length (mm TL) and weight (whole) at age
meanlen_TL=xmean(len_probe(time,meanlen_TL,vgtpar;b)); //total length in mm
wgt_vgtpar=vgtpar(meanlen_TL,vgtpar;b); //whole wgt in kg
wgt_vgtpar=vgtpar(meanlen_TL,vgtpar;b); //convert wgt in kg to weight in g
wgt_vgtpar=vgtpar(meanlen_TL,vgtpar;b); //convert weight in g to weight in mt
wgt_vgtpar=vgtpar(meanlen_TL,vgtpar;b); //lbf of weight mt
wgt_vgtpar=vgtpar(meanlen_TL,vgtpar;b); //lbf of weight lbf

FUNCTION get_reprod
//reprod is the product of stuff going into reproductive capacity calc;
for (iyear=stryr; iyear=endyr; iyear++)
{
  reprod(iyear)=elem_prod((elem_prod(prop_f(iyear),maturity_f)+elem_prod((prop_m(iyear)),maturity_m)),wgt_mt); //both sexes
  reprod2(iyear)=elem_prod(elem_prod(prop_f(iyear),maturity_f),wgt_mt); //females only
}

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);
  cprob_lenvec(nlenbins)=1.0-cumd_norm(zscore_len);
  lenprob(iage,nlenbins)=1.0-cprob_lenvec(nlenbins); //renormalize to account for any prob mass below size=0
}
//fleet and survey specific length probs, all assumed here to equal the popn
lenprob_cH=lenprob;
lenprob_cD=lenprob;
lenprob_HB=lenprob;

FUNCTION get_weight_at_age_landings
for (iyear=stryr; iyear=endyr; iyear++)
{
  len_cH_mm(iyear)=meanlen_TL;
  gutwgt_cH_klb(iyear)=wgt_klb_gut;
  wholewgt_cH_klb(iyear)=wgt_klb;
  len_cD_mm(iyear)=meanlen_TL;
  gutwgt_cD_klb(iyear)=wgt_klb_gut;
  len_HB_mm(iyear)=meanlen_TL;
  gutwgt_HB_klb(iyear)=wgt_klb_gut;
  len_GR_mm(iyear)=meanlen_TL;
  gutwgt_GR_klb(iyear)=wgt_klb_gut;
  len_cH_D_mm(iyear)=meanlen_TL;
  gutwgt_cH_D_klb(iyear)=wgt_klb_gut;
  len_HB_D_mm(iyear)=meanlen_TL;
  gutwgt_HB_D_klb(iyear)=wgt_klb_gut;
  len_GR_D_mm(iyear)=meanlen_TL;
  gutwgt_GR_D_klb(iyear)=wgt_klb_gut;
}

FUNCTION get_spr_F0
//at mdyr, apply half this yr's natural 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(endyr)));
bpr_F0=sum(elem_prod(N_bpr_F0,wgt_mt));
FUNCTION get_selectivity

//BLOCK 1 for selex.
for (iyear=styr; iyear<=endyr_selex_phase1; iyear++)
{
  sel_cH(iyear)=logistic(agebins, selpar_L50_cH1, selpar_slope_cH1);
  sel_cD(iyear)=logistic_exponential(agebins, selpar_L50_cD, selpar_slope_cD, selpar_sigma_cD, selpar_afull_cD);
  sel_HB(iyear)=logistic(agebins, selpar_L50_HB1, selpar_slope_HB1);
  sel_cD(iyear)(13,nages)=sel_cD(iyear)(12);
}

//BLOCK 2 for selex.
for (iyear=(endyr_selex_phase1+1); iyear<=endyr_selex_phase2; iyear++)
{
  sel_cH(iyear)=logistic(agebins, selpar_L50_cH2, selpar_slope_cH2);
  sel_cD(iyear)=logistic_exponential(agebins, selpar_L50_cD, selpar_slope_cD, selpar_sigma_cD, selpar_afull_cD);
  sel_HB(iyear)=logistic(agebins, selpar_L50_HB2, selpar_slope_HB2);
  sel_cD(iyear)(13,nages)=sel_cD(iyear)(12);
}

//BLOCK 3 for selex.
for (iyear=(endyr_selex_phase2+1); iyear<=endyr; iyear++)
{
  sel_cH(iyear)=logistic(agebins, selpar_L50_cH3, selpar_slope_cH3);
  sel_cD(iyear)=logistic_exponential(agebins, selpar_L50_cD, selpar_slope_cD, selpar_sigma_cD, selpar_afull_cD);
  sel_HB(iyear)=logistic(agebins, selpar_L50_HB3, selpar_slope_HB3);
  sel_cD(iyear)(13,nages)=sel_cD(iyear)(12);
}

//Discard selex, uses a 2 age shift
for (iyear=styr_HB_D;iyear<=endyr_HB_D;iyear++)
{
  for (iage=1; iage<=(nages-2); iage++)
  {
    sel_HB_D(iyear,iage)=(sel_HB(endyr,iage+2)-sel_HB(endyr,iage));
    if(sel_HB_D(iyear,iage)<0.0) {sel_HB_D(iyear,iage)=0.0;}
  }
  sel_HB_D(iyear,(nages-1))=0.0;
  sel_HB_D(iyear,nages)=0.0;
  sel_HB_D(iyear)=sel_HB_D(iyear)/max(sel_HB_D(iyear));
}

for (iyear=styr_cH_D;iyear<=endyr_cH_D;iyear++)
{
  for (iage=1; iage<=(nages-2); iage++)
  {
    sel_cH_D(iyear,iage)=(sel_cH(endyr,iage+2)-sel_cH(endyr,iage));
    if(sel_cH_D(iyear,iage)<0.0) {sel_cH_D(iyear,iage)=0.0;}
  }
  sel_cH_D(iyear,(nages-1))=0.0;
  sel_cH_D(iyear,nages)=0.0;
  sel_cH_D(iyear)=sel_cH_D(iyear)/max.sel_cH_D(iyear));
}

FUNCTION get_mortality

Fsum.initialize();
Fapex.initialize();
F.initialize();
//initialization F is avg from first 3 yrs of observed landings: cD excluded bc it begins later than styr
log_F_dev_init_cH=sum(log_F_dev_cH(styr_cH_L,(styr_cH_L+2)))/3.0;
log_F_dev_init_HB=sum(log_F_dev_HB(styr_HB_L,(styr_HB_L+2)))/3.0;
log_F_dev_init_GR=sum(log_F_dev_GR(styr_GR_L,(styr_GR_L+2)))/3.0;
for (iyear=styr; iyear<=endyr; 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_cD_L & iyear<=endyr_cD_L)
  { F_cD_out(iyear)=mfexp(log_avg_F_cD+log_F_dev_cD(iyear));
  F_cD(iyear)=sel_cD(iyear)*F_cD_out(iyear);
  Fsum(iyear)+=F_cD_out(iyear);
  }
  if(iyear>=styr_HB_L & iyear<=endyr_HB_L)
  { F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_dev_HB(iyear));
  F_HB(iyear)=sel_HB(iyear)*F_HB_out(iyear);
  Fsum(iyear)+=F_HB_out(iyear);
  }
  if(iyear>=styr_GR_L & iyear<=endyr_GR_L)
  { F_GR_out(iyear)=mfexp(log_avg_F_GR+log_F_dev_GR(iyear));
  F_GR(iyear)=sel_HB(iyear)*F_GR_out(iyear); //general rec shares headboat selex
  Fsum(iyear)+=F_GR_out(iyear);
  }
  if(iyear>=styr_cH_D & iyear<=endyr_cH_D)
  { F_cH_D_out(iyear)=mfexp(log_avg_F_cH_D+log_F_dev_cH_D(iyear));
  F_cH_D(iyear)=sel_cH_D(iyear)*F_cH_D_out(iyear);
  Fsum(iyear)+=F_cH_D_out(iyear);
  }
  if(iyear>=styr_HB_D & iyear<=endyr_HB_D)
  { F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_dev_HB_D(iyear));
  F_HB_D(iyear)=sel_HB_D(iyear)*F_HB_D_out(iyear);
  Fsum(iyear)+=F_HB_D_out(iyear);
  }
Fsum(iyear)+=F_HB_D_out(iyear);

if(iyear>=styr_GR_D & iyear<=endyr_GR_D)
{
 F_GR_D_out(iyear)=mfexp(log_avg_F_GR_D+log_F_dev_GR_D(iyear));
 F_GR_D(iyear)=sel_HB_D(iyear)*F_GR_D_out(iyear); //general rec shares headboat selex
 Fsum(iyear)+=F_GR_D_out(iyear);
}

//Total F at age
F(iyear)=F_cH(iyear); //first in additive series (NO +=)
F(iyear)+=F_cD(iyear);
F(iyear)+=F_HB(iyear);
F(iyear)+=F_GR(iyear);
F(iyear)+=F_cH_D(iyear);
F(iyear)+=F_HB_D(iyear);
F(iyear)+=F_GR_D(iyear);
Fapex(iyear)=max(F(iyear));
Z(iyear)=M+F(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_denom=mfexp(log_avg_F_cH+log_F_dev_init_cH)+mfexp(log_avg_F_HB+log_F_dev_init_HB)+mfexp(log_avg_F_GR+log_F_dev_init_GR);
F_init_cH_prop=mfexp(log_avg_F_cH+log_F_dev_init_cH)/F_init_denom;
F_init_HB_prop=mfexp(log_avg_F_HB+log_F_dev_init_HB)/F_init_denom;
F_init_GR_prop=mfexp(log_avg_F_GR+log_F_dev_init_GR)/F_init_denom;
F_initial=sel_cH(styr)*F_init*F_init_cH_prop+
 sel_HB(styr)*F_init*F_init_HB_prop+
 sel_HB(styr)*F_init*F_init_GR_prop; //GR uses HB selex
Z_initial=M+F_initial;

//Initial equilibrium age structure
N_initial_eq(1)=R1; //at peak spawning time;
for (iage=2; iage<=nages; iage++)
{
 N_initial_eq(iage)=N_initial_eq(iage-1)*
 mfexp(-1.0*(Z_initial(iage-1)));
}
N_initial_eq(nages)=N_initial_eq(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group calculation
N_initial_eq(nages)=wlems_prod(N_initial_eq(nages), set_q_DD_stage,nages);

//Add deviations to initial equilibrium N
H(styr)(set_q_DD_stage,nages)=wlems_prod(H(styr), set_q_DD_stage,nages);
N(styr)(set_q_DD_stage,nages)=wlems_prod(N(styr), set_q_DD_stage,nages);

SSB(styr)=sum(elem_prod(N_styr, reprod(styr)));
MatFemB(styr)=sum(elem_prod(N_styr, reprod2(styr)));
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

 else //recruitment follows S-R curve with lognormal deviation

FUNCTION get_landings_numbers //Baranov catch eqn
for (iyear=styr; iyear<=endyr; iyear++)
{
  for (iage=1; iage<=nages; iage++)
  {
    L_cH_num(iyear,iage)=N(iyear,iage)*F_cH(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    L_cD_num(iyear,iage)=N(iyear,iage)*F_cD(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    L_HB_num(iyear,iage)=N(iyear,iage)*F_HB(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    L_GR_num(iyear,iage)=N(iyear,iage)*F_GR(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
  }
  pred_cH_L_knum(iyear)=sum(L_cH_num(iyear))/1000.0;
  pred_cD_L_knum(iyear)=sum(L_cD_num(iyear))/1000.0;
  pred_HB_L_knum(iyear)=sum(L_HB_num(iyear))/1000.0;
  pred_GR_L_knum(iyear)=sum(L_GR_num(iyear))/1000.0;
}

FUNCTION get_landings_wgt
for (iyear=styr; iyear<=endyr; iyear++)
{
  for (iage=1; iage<=nages; iage++)
  {
    L_cH_klb(iyear)=elem_prod(L_cH_num(iyear),gutwgt_cH_klb(iyear)); //in 1000 lb gutted weight
    L_cD_klb(iyear)=elem_prod(L_cD_num(iyear),gutwgt_cD_klb(iyear)); //in 1000 lb gutted weight
    L_HB_klb(iyear)=elem_prod(L_HB_num(iyear),gutwgt_HB_klb(iyear)); //in 1000 lb gutted weight
    L_GR_klb(iyear)=elem_prod(L_GR_num(iyear),gutwgt_GR_klb(iyear)); //in 1000 lb gutted weight
  }
  pred_cH_L_klb(iyear)=sum(L_cH_klb(iyear));
  pred_cD_L_klb(iyear)=sum(L_cD_klb(iyear));
  pred_HB_L_klb(iyear)=sum(L_HB_klb(iyear));
  pred_GR_L_klb(iyear)=sum(L_GR_klb(iyear));
}

FUNCTION get_dead_discards
//dead discards at age (number fish)
for (iyear=styr_cH_D; iyear<=endyr_cH_D; iyear++)
{
  for (iage=1; iage<=nages; iage++)
  {
    D_cH_num(iyear,iage)=N(iyear,iage)*F_cH_D(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
  }
  pred_cH_D_knum(iyear)=sum(D_cH_num(iyear))/1000.0; //pred annual dead discards in 1000s (for matching data)
  pred_cH_D_klb(iyear)=sum(elem_prod(D_cH_num(iyear),gutwgt_cH_D_klb(iyear))); //annual dead discards in 1000 lb gutted (for output only)
}
for (iyear=styr_HB_D; iyear<=endyr_HB_D; iyear++)
{
  for (iage=1; iage<=nages; iage++)
  {
    D_HB_num(iyear,iage)=N(iyear,iage)*F_HB_D(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
  }
  pred_HB_D_knum(iyear)=sum(D_HB_num(iyear))/1000.0; //pred annual dead discards in 1000s (for matching data)
  pred_HB_D_klb(iyear)=sum(elem_prod(D_HB_num(iyear),gutwgt_HB_D_klb(iyear))); //annual dead discards in 1000 lb gutted (for output only)
}
for (iyear=styr_GR_D; iyear<=endyr_GR_D; iyear++)
{
  for (iage=1; iage<=nages; iage++)
  {
    D_GR_num(iyear,iage)=N(iyear,iage)*F_GR_D(iyear,iage)*
      (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
  }
  pred_GR_D_knum(iyear)=sum(D_GR_num(iyear))/1000.0; //pred annual dead discards in 1000s (for matching data)
  pred_GR_D_klb(iyear)=sum(elem_prod(D_GR_num(iyear),gutwgt_GR_D_klb(iyear))); //annual dead discards in 1000 lb gutted (for output only)
}

FUNCTION get_catchability_fcns
//Get rate increase if estimated, otherwise fixed above
if (set_q_rate_phase>0.0)
{
  for (iyear=styr_cH_cpue; iyear<=endyr_cH_cpue; iyear++)
  {
    if (iyear>styr_cH_cpue & iyear <=2003)
    {
      q_rate_fcn_cH(iyear)=(1.0+(iyear-styr_cH_cpue)*q_rate)*q_rate_fcn_cH(styr_cH_cpue); //linear
    }
    else
    {
      q_rate_fcn_cH(iyear)=q_rate_fcn_cH(iyear-1); //compound
    }
  }
  for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
  {
    if (iyear>styr_HB_cpue & iyear <=2003)
    {
      q_rate_fcn_HB(iyear)=(1.0+(iyear-styr_HB_cpue)*q_rate)*q_rate_fcn_HB(styr_HB_cpue); //linear
    }
    else
    {
      q_rate_fcn_HB(iyear)=q_rate_fcn_HB(iyear-1); //compound
    }
  }
  for (iyear=styr_GR_cpue; iyear<=endyr_GR_cpue; iyear++)
  {
    if (iyear>styr_GR_cpue & iyear <=2003)
    {
      q_rate_fcn_GR(iyear)=(1.0+(iyear-styr_GR_cpue)*q_rate)*q_rate_fcn_GR(styr_GR_cpue); //linear
    }
    else
    {
      q_rate_fcn_GR(iyear)=q_rate_fcn_GR(iyear-1); //compound
    }
  }
}
if (iyear>2003) {q_rate_fcn_HB(iyear)=q_rate_fcn_HB(iyear-1);}
for (iyear=styr_GR_cpue; iyear<=endyr_GR_cpue; iyear++)
{ if (iyear>styr_GR_cpue & iyear <=2003)
{ //q_rate_fcn_GR(iyear)=(1.0+q_rate)*q_rate_fcn_GR(iyear-1); //compound
q_rate_fcn_GR(iyear)=(1.0+(iyear-styr_GR_cpue)*q_rate)*q_rate_fcn_GR(styr_GR_cpue); //linear
}
if (iyear>2003) {q_rate_fcn_GR(iyear)=q_rate_fcn_GR(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------------------------
//cH cpue
q_cH(styr_cH_cpue)=mfexp(log_q_cH);
for (iyear=styr_cH_cpue; iyear<=endyr_cH_cpue; iyear++)
{
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));}
}

//HB cpue
q_HB(styr_HB_cpue)=mfexp(log_q_HB);
for (iyear=styr_HB_cpue; iyear<=endyr_HB_cpue; iyear++)
{
N_HB(iyear)=elem_prod(N_mdyr(iyear),sel_HB(iyear));
pred_HB_cpue(iyear)=q_HB(iyear)*q_rate_fcn_HB(iyear)*q_DD_fcn(iyear)*sum(N_HB(iyear));
if (iyear<endyr_HB_cpue){q_HB(iyear+1)=q_HB(iyear)*mfexp(q_RW_log_dev_HB(iyear));}
}

//GR cpue
q_GR(styr_GR_cpue)=mfexp(log_q_GR);
for (iyear=styr_GR_cpue; iyear<=endyr_GR_cpue; iyear++)
{
N_GR(iyear)=elem_prod(N_mdyr(iyear),sel_HB(iyear)); //GR uses HB selex
pred_GR_cpue(iyear)=q_GR(iyear)*q_rate_fcn_GR(iyear)*q_DD_fcn(iyear)*sum(N_GR(iyear));
if (iyear<endyr_GR_cpue){q_GR(iyear+1)=q_GR(iyear)*mfexp(q_RW_log_dev_GR(iyear));}
}

FUNCTION get_length_comps

//commercial handline
for (iyear=1;iyear<=nyr_cH_lenc;iyear++)
{pred_cH_lenc(iyear)=(L_cH_num(yrs_cH_lenc(iyear))*lenprob_cH)/sum(L_cH_num(yrs_cH_lenc(iyear)));
}
//commercial diving
for (iyear=1;iyear<=nyr_cD_lenc;iyear++)
{pred_cD_lenc(iyear)=(L_cD_num(yrs_cD_lenc(iyear))*lenprob_cD)/sum(L_cD_num(yrs_cD_lenc(iyear)));
}
//headboat
for (iyear=1;iyear<=nyr_HB_lenc;iyear++)
{pred_HB_lenc(iyear)=(L_HB_num(yrs_HB_lenc(iyear))*lenprob_HB)/sum(L_HB_num(yrs_HB_lenc(iyear)));
}

FUNCTION get_age_comps

//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);}
}

//commercial diving
for (iyear=1;iyear<=nyr_cD_agec;iyear++)
{
ErrorFree_cD_agec(iyear)=L_cD_num(yrs_cD_agec(iyear))/sum(L_cD_num(yrs_cD_agec(iyear)));
pred_cD_agec_allages(iyear)=age_error*(ErrorFree_cD_agec(iyear)/sum(ErrorFree_cD_agec(iyear)));
for (iage=1; iage<=nages_agec; iage++) {pred_cD_agec(iyear,iage)=pred_cD_agec_allages(iyear,iage);}
for (iage=(nages_agec+1); iage<=nages; iage++) {pred_cD_agec(iyear,nages_agec)+=pred_cD_agec_allages(iyear,iage);}
}

//headboat
for (iyear=1;iyear<=nyr_HB_agec;iyear++)
{
ErrorFree_HB_agec(iyear)=L_HB_num(yrs_HB_agec(iyear))/sum(L_HB_num(yrs_HB_agec(iyear)));
pred_HB_agec_allages(iyear)=age_error*ErrorFree_HB_agec(iyear);
for (iage=1; iage<=nages_agec; iage++) {pred_HB_agec(iyear,iage)=pred_HB_agec_allages(iyear,iage);}
for (iage=(nages_agec+1); iage<=nages; iage++) {pred_HB_agec(iyear,nages_agec)+=pred_HB_agec_allages(iyear,iage);}
}
\[
\begin{align*}
\text{sum}(\log_F_{\text{dev}}_{cD}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_temp_sum} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{\text{HB}} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{\text{HB}}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_temp_sum} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{\text{GR}} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{\text{GR}}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_temp_sum} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{cH_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{cH_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_temp_sum} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{HB_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{HB_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_temp_sum} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{GR_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{GR_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_cH_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{cH} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{cH}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_cD_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{cD} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{cD}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_HB_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{HB} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{HB}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_GR_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{GR} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{GR}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_cH_D_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{cH_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{cH_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_HB_D_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{HB_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{HB_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\text{F_GR_D_prop} &= \text{mfexp}(\text{selpar}_n_yrs_wgted \cdot \log_{\text{avg}}_F_{GR_D} + \\
& \quad \text{sum}(\log_F_{\text{dev}}_{GR_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted) / \text{F_temp_sum}; \\
\log_F_{\text{dev}}_{end\_cH} &= \text{sum}(\log_F_{\text{dev}}_{cH}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_cD} &= \text{sum}(\log_F_{\text{dev}}_{cD}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_HB} &= \text{sum}(\log_F_{\text{dev}}_{HB}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_GR} &= \text{sum}(\log_F_{\text{dev}}_{GR}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_cH_D} &= \text{sum}(\log_F_{\text{dev}}_{cH_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_HB_D} &= \text{sum}(\log_F_{\text{dev}}_{HB_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\log_F_{\text{dev}}_{end\_GR_D} &= \text{sum}(\log_F_{\text{dev}}_{GR_D}(\text{endyr}-\text{selpar}_n_yrs_wgted+1, \text{endyr})) / \text{selpar}_n_yrs_wgted; \\
\text{F_end\_L} &= \text{sel\_cH}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{cH} + \log_F_{\text{dev}}_{end\_cH}) + \\
& \quad \text{sel\_cD}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{cD} + \log_F_{\text{dev}}_{end\_cD}) + \\
& \quad \text{sel\_HB}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{HB} + \log_F_{\text{dev}}_{end\_HB}) + \\
& \quad \text{sel\_GR}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{GR} + \log_F_{\text{dev}}_{end\_GR}); //GR uses HB selex \\
\text{F_end\_D} &= \text{sel\_cH\_D}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{cH\_D} + \log_F_{\text{dev}}_{end\_cH\_D}) + \\
& \quad \text{sel\_HB\_D}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{HB\_D} + \log_F_{\text{dev}}_{end\_HB\_D}) + \\
& \quad \text{sel\_HB\_D}(\text{endyr}) \cdot \text{mfexp}(\log_{\text{avg}}_F_{GR\_D} + \log_F_{\text{dev}}_{end\_GR\_D}); //GR uses HB selex
\end{align*}
\]
\( B_{eq}(ff) = \sum(\text{elem}_{prod}(N_{age\_msy}, wgt_{mt})) \); // in gutted weight
\( L_{eq\_knum}(ff) = \frac{\sum(\text{elem}_{prod}(L_{age\_msy}, wgt_{wgted\_L\_klb}))}{1000.0}; \)
\( D_{eq\_knum}(ff) = \frac{\sum(D_{age\_msy})}{1000.0}; \)
\( \text{msy}_{klb\_out} = \max(L_{eq\_klb}); \) // msy in gutted weight

for (ff=1; ff<=n_iter_msy; ff++) {
    if (L_{eq\_klb}(ff) == \text{msy}_{klb\_out}) {
        \( \text{SSB}_{msy\_out} = \text{SSB}_{eq}(ff); \)
        \( B_{msy\_out} = B_{eq}(ff); \)
        \( R_{msy\_out} = R_{eq}(ff); \)
        \( \text{msy}_{knum\_out} = L_{eq\_knum}(ff); \)
        \( D_{msy\_knum\_out} = D_{eq\_knum}(ff); \)
        \( D_{msy\_klb\_out} = D_{eq\_klb}(ff); \)
        \( F_{msy\_out} = F_{msy}(ff); \)
        \( \text{spr}_{msy\_out} = \text{spr}_{msy}(ff); \)
    }
}

FUNCTION get_miscellaneous_stuff
// switch here if var_rec_dev <= dzero
if (var_rec_dev > 0.0) {
    \( \sigma_{rec\_dev} = \sqrt{\text{var}_{rec\_dev}}; \)
} else {
    \( \sigma_{rec\_dev} = 0.0; \)
}
\( \text{len}_{cv} = \frac{\text{elem}_{div}(\text{len}_{sd}, \text{meanlen}_{TL})}{1000}; \)

// compute total landings- and discards-at-age in 1000 fish and klb gutted weight
\( \text{L\_total\_num\_initialized}; \)
\( \text{L\_total\_klb\_initialized}; \)
\( \text{L\_total\_knum\_yr\_initialized}; \)
\( \text{L\_total\_klb\_yr\_initialized}; \)
\( \text{D\_total\_num\_initialized}; \)
\( \text{D\_total\_klb\_initialized}; \)
\( \text{D\_total\_knum\_yr\_initialized}; \)
\( \text{D\_total\_klb\_yr\_initialized}; \)
\( \text{D\_cH\_klb\_initialized}; \)
\( \text{D\_HB\_klb\_initialized}; \)
\( \text{D\_GR\_klb\_initialized}; \)

for (iyear = styr; iyear <= endyr; iyear++) {
    \( \text{L\_total\_klb\_yr}(iyear) = \text{pred}_{cH\_L\_klb}(iyear) + \text{pred}_{cD\_L\_klb}(iyear) + \text{pred}_{HB\_L\_klb}(iyear) + \text{pred}_{GR\_L\_klb}(iyear); \)
    \( \text{L\_total\_knum\_yr}(iyear) = \text{pred}_{cH\_L\_knum}(iyear) + \text{pred}_{cD\_L\_knum}(iyear) + \text{pred}_{HB\_L\_knum}(iyear) + \text{pred}_{GR\_L\_knum}(iyear); \)
    \( B(iyear) = \sum(\text{N}(iyear) \times \text{wgt}_{mt}); \)
    \( \text{totN}(iyear) = \sum(\text{N}(iyear)); \)
    \( \text{totB}(iyear) = \sum(\text{B}(iyear)); \)
    if (iyear >= styr_cH_D && iyear <= endyr_cH_D) {
        \( \text{D\_total\_knum\_yr}(iyear) += \text{pred}_{cH\_D\_knum}(iyear); \)
        \( \text{D\_total\_klb\_yr}(iyear) += \text{pred}_{cH\_D\_klb}(iyear); \)
        \( \text{D\_cH\_klb}(iyear) = \sum(\text{D\_cH\_num}(iyear) \times \text{gutwgt}_{cH\_D\_klb}(iyear)); \) // in 1000 lb gutted
    }
    if (iyear >= styr_HB_D && iyear <= endyr_HB_D) {
        \( \text{D\_total\_knum\_yr}(iyear) += \text{pred}_{HB\_D\_knum}(iyear); \)
        \( \text{D\_total\_klb\_yr}(iyear) += \text{pred}_{HB\_D\_klb}(iyear); \)
        \( \text{D\_HB\_klb}(iyear) = \sum(\text{D\_HB\_num}(iyear) \times \text{gutwgt}_{HB\_D\_klb}(iyear)); \) // in 1000 lb gutted
    }
    if (iyear >= styr_GR_D && iyear <= endyr_GR_D) {
        \( \text{D\_total\_knum\_yr}(iyear) += \text{pred}_{GR\_D\_knum}(iyear); \)
        \( \text{D\_total\_klb\_yr}(iyear) += \text{pred}_{GR\_D\_klb}(iyear); \)
        \( \text{D\_GR\_klb}(iyear) = \sum(\text{D\_GR\_num}(iyear) \times \text{gutwgt}_{GR\_D\_klb}(iyear)); \) // in 1000 lb gutted
    }
}
\( \text{L\_total\_num\_D\_cH\_num\_normalized}; \) // landings at age in number fish
\( \text{L\_total\_klb\_D\_cH\_klb\_normalized}; \) // landings at age in klb gutted weight
\( \text{D\_total\_num\_D\_cH\_num\_normalized}; \) // discards at age in number fish
\( \text{D\_total\_klb\_D\_cH\_klb\_normalized}; \) // discards at age in klb gutted weight

// Time series of interest
\( \text{N}(endyr) = \sum(\text{N}(endyr) \times \text{wgt}_{mt}); \)
\( \text{totN}(endyr) = \sum(\text{N}(endyr)); \)
\( \text{sS}(endyr) = \sum(\text{sS}(endyr)); \)
\( \text{SSB}(endyr) = \sum(\text{elem}_{prod}(\text{N\_spawn}(endyr) \times \text{reprod}(endyr))); \)
\( \text{MatFemB}(endyr) = \sum(\text{elem}_{prod}(\text{N\_spawn}(endyr) \times \text{reprod2}(endyr))); \)
\( \text{rec}(\text{N}(endyr)); \)
\( \text{SSD}(\text{SSB}(endyr)); \)
\( \text{F}_{\text{may\_out}}(0); \)
\( \text{F}_{\text{may\_end}}(\text{F}_{\text{may\_out}}(0)); \)
\( \text{F}_{\text{may\_end}}(\text{F}_{\text{may\_end}}(\text{F}_{\text{may\_out}}(0)) \times \text{F}_{\text{may\_out}}(0)); \)
\( \text{if}(\text{SSB\_may\_out}(0)) \)
    \( \text{SSD}_{\text{SSB\_may\_out}}(\text{SSB\_may\_out}(0)); \)
}
// 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(iyear)))/spr_F0;
        }

FUNCTION get_effective_sample_sizes
    neff_cH_lenc_allyr_out=missing;
    neff_cD_lenc_allyr_out=missing;
    neff_HB_lenc_allyr_out=missing;
    neff_cH_agec_allyr_out=missing;
    neff_cD_agec_allyr_out=missing;
    neff_HB_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_cD_lenc; iyear++)
        {if (nsamp_cD_lenc(iyear)>=minSS_cD_lenc)
            {neff_cD_lenc_allyr_out(yrs_cD_lenc(iyear))=multinom_eff_N(pred_cD_lenc(iyear),obs_cD_lenc(iyear));}
        else {neff_cD_lenc_allyr_out(yrs_cD_lenc(iyear))=-99;}}
    for (iyear=1; iyear<=nyr_HB_lenc; iyear++)
        {if (nsamp_HB_lenc(iyear)>=minSS_HB_lenc)
            {neff_HB_lenc_allyr_out(yrs_HB_lenc(iyear))=multinom_eff_N(pred_HB_lenc(iyear),obs_HB_lenc(iyear));}
        else {neff_HB_lenc_allyr_out(yrs_HB_lenc(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_cD_agec; iyear++)
        {if (nsamp_cD_agec(iyear)>=minSS_cD_agec)
            {neff_cD_agec_allyr_out(yrs_cD_agec(iyear))=multinom_eff_N(pred_cD_agec(iyear),obs_cD_agec(iyear));}
        else {neff_cD_agec_allyr_out(yrs_cD_agec(iyear))=-99;}}
    for (iyear=1; iyear<=nyr_HB_agec; iyear++)
        {if (nsamp_HB_agec(iyear)>=minSS_HB_agec)
            {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=multinom_eff_N(pred_HB_agec(iyear),obs_HB_agec(iyear));}
        else {neff_HB_agec_allyr_out(yrs_HB_agec(iyear))=-99;}}
FUNCTION evaluate_objective_function
fval=0.0;
fval_data=0.0;

//---Indices-----------------------------
f_cH_cpue=0.0;
f_cH_cpue=lk_lognormal(pred_cH_cpue, obs_cH_cpue, cH_cpue_cv, w_I_cH);
fval+=f_cH_cpue;
fval_data+=f_cH_cpue;

f_cH_L=0.0;
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_cD_cpue=0.0;
f_cD_cpue=lk_lognormal(pred_cD_cpue, obs_cD_cpue, cD_cpue_cv, w_I_cD);
fval+=f_cD_cpue;
fval_data+=f_cD_cpue;

f_cD_L=0.0;
f_cD_L=lk_lognormal(pred_cD_L_klb(styr_cD_L,endyr_cD_L), obs_cD_L(styr_cD_L,endyr_cD_L),
cD_L_cv(styr_cD_L,endyr_cD_L), w_L);
fval+=f_cD_L;
fval_data+=f_cD_L;

f_HB_cpue=0.0;
f_HB_cpue=lk_lognormal(pred_HB_cpue, obs_HB_cpue, HB_cpue_cv, w_I_HB);
fval+=f_HB_cpue;
fval_data+=f_HB_cpue;

f_HB_L=0.0;
f_HB_L=lk_lognormal(pred_HB_L_knum(styr_HB_L,endyr_HB_L), obs_HB_L(styr_HB_L,endyr_HB_L),
HB_L_cv(styr_HB_L,endyr_HB_L), w_L);
fval+=f_HB_L;
fval_data+=f_HB_L;

f_GR_cpue=0.0;
f_GR_cpue=lk_lognormal(pred_GR_cpue, obs_GR_cpue, GR_cpue_cv, w_I_GR);
fval+=f_GR_cpue;
fval_data+=f_GR_cpue;

f_GR_L=0.0;
f_GR_L=lk_lognormal(pred_GR_L_knum(styr_GR_L,endyr_GR_L), obs_GR_L(styr_GR_L,endyr_GR_L),
GR_L_cv(styr_GR_L,endyr_GR_L), w_L);
fval+=f_GR_L;
fval_data+=f_GR_L;

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

f_cH_L in 1000 lb gutted wt
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_cD_L in 1000 lb gutted wt
f_cD_L=lk_lognormal(pred_cD_L_klb(styr_cD_L,endyr_cD_L), obs_cD_L(styr_cD_L,endyr_cD_L),
cD_L_cv(styr_cD_L,endyr_cD_L), w_L);
fval+=f_cD_L;
fval_data+=f_cD_L;

f_HB_L in 1000 fish
f_HB_L=lk_lognormal(pred_HB_L_knum(styr_HB_L,endyr_HB_L), obs_HB_L(styr_HB_L,endyr_HB_L),
HB_L_cv(styr_HB_L,endyr_HB_L), w_L);
fval+=f_HB_L;
fval_data+=f_HB_L;

f_GR_L in 1000 fish
f_GR_L=lk_lognormal(pred_GR_L_knum(styr_GR_L,endyr_GR_L), obs_GR_L(styr_GR_L,endyr_GR_L),
GR_L_cv(styr_GR_L,endyr_GR_L), w_L);
fval+=f_GR_L;
fval_data+=f_GR_L;

//---Discards---------------------------

f_cH_D in 1000 fish
f_cH_D=lk_lognormal(pred_cH_D_knum(styr_cH_D,endyr_cH_D), obs_cH_D(styr_cH_D,endyr_cH_D),
cH_D_cv(styr_cH_D,endyr_cH_D), w_D);
fval+=f_cH_D;
fval_data+=f_cH_D;

f_HB_D in 1000 fish
f_HB_D=lk_lognormal(pred_HB_D_knum(styr_HB_D,endyr_HB_D), obs_HB_D(styr_HB_D,endyr_HB_D),
HB_D_cv(styr_HB_D,endyr_HB_D), w_D);
fval+=f_HB_D;
fval_data+=f_HB_D;

f_GR_D in 1000 fish
f_GR_D=lk_lognormal(pred_GR_D_knum(styr_GR_D,endyr_GR_D), obs_GR_D(styr_GR_D,endyr_GR_D),
GR_D_cv(styr_GR_D,endyr_GR_D), w_D);
fval+=f_GR_D;
fval_data+=f_GR_D;

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

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

f_cD_lenc=lk_robust_multinomial(nsamp_cD_lenc, pred_cD_lenc, obs_cD_lenc, nyr_cD_lenc, double(nlenbins), minSS_cD_lenc, u_l_cD);
fval+=f_cD_lenc;
fval_data+=f_cD_lenc;

f_HB_lenc=lk_robust_multinomial(nsamp_HB_lenc, pred_HB_lenc, obs_HB_lenc, nyr_HB_lenc, double(nlenbins), minSS_HB_lenc, u_l_HB);
fval+=f_HB_lenc;
fval_data+=f_HB_lenc;

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

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

f_cD_agec=lk_robust_multinomial(nsamp_cD_agec, pred_cD_agec, obs_cD_agec, nyr_cD_agec, double(nages_agec), minSS_cD_agec, u_ac_cD);
fval+=f_cD_agec;
fval_data+=f_cD_agec;

f_HB_agec
f_HB_agec谈到robust_multinomial(nsamp_HB_agec, pred_HB_agec, obs_HB_agec, nyr_HB_agec, double(nages_agec), minSS_HB_agec, w_ac_HB);
fval=f_HB_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));}
  }
}

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;
}

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);}}
}

f_priors=0.0;
for (iyear=styr; iyear<endyr_HB_cpue; iyear++)
{f_HB_RW_cpue+=square(q_RW_log_dev_HB(iyear))/(2.0*set_q_RW_HB_var);} 
fval+=f_HB_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+=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_cH1,set_selpar_L50_cH1(5), set_selpar_L50_cH1(6), set_selpar_L50_cH1(7));
f_priors+=neg_log_prior(selpar_slope_cH1,set_selpar_slope_cH1(5), set_selpar_slope_cH1(6), set_selpar_slope_cH1(7));
f_priors+=neg_log_prior(selpar_L50_cH2,set_selpar_L50_cH2(5), set_selpar_L50_cH2(6), set_selpar_L50_cH2(7));
f_priors+=neg_log_prior(selpar_slope_cH2,set_selpar_slope_cH2(5), set_selpar_slope_cH2(6), set_selpar_slope_cH2(7));
f_priors+=neg_log_prior(selpar_L50_cH3,set_selpar_L50_cH3(5), set_selpar_L50_cH3(6), set_selpar_L50_cH3(7));
f_priors+=neg_log_prior(selpar_slope_cH3,set_selpar_slope_cH3(5), set_selpar_slope_cH3(6), set_selpar_slope_cH3(7));
f_priors+=neg_log_prior(selpar_L50_cD,set_selpar_L50_cD(5), set_selpar_L50_cD(6), set_selpar_L50_cD(7));
f_priors+=neg_log_prior(selpar_slope_cD,set_selpar_slope_cD(5), set_selpar_slope_cD(6), set_selpar_slope_cD(7));
f_priors+=neg_log_prior(selpar_afull_cD,set_selpar_afull_cD(5), set_selpar_afull_cD(6), set_selpar_afull_cD(7));
f_priors+=neg_log_prior(selpar_sigma_cD,set_selpar_sigma_cD(5), set_selpar_sigma_cD(6), set_selpar_sigma_cD(7));
f_priors+=neg_log_prior(selpar_L50_HB1,set_selpar_L50_HB1(5), set_selpar_L50_HB1(6), set_selpar_L50_HB1(7));
f_priors+=neg_log_prior(selpar_slope_HB1,set_selpar_slope_HB1(5), set_selpar_slope_HB1(6), set_selpar_slope_HB1(7));
f_priors+=neg_log_prior(selpar_L50_HB2,set_selpar_L50_HB2(5), set_selpar_L50_HB2(6), set_selpar_L50_HB2(7));
f_priors+=neg_log_prior(selpar_slope_HB2,set_selpar_slope_HB2(5), set_selpar_slope_HB2(6), set_selpar_slope_HB2(7));
f_priors+=neg_log_prior(selpar_L50_HB3,set_selpar_L50_HB3(5), set_selpar_L50_HB3(6), set_selpar_L50_HB3(7));
f_priors+=neg_log_prior(selpar_slope_HB3,set_selpar_slope_HB3(5), set_selpar_slope_HB3(6), set_selpar_slope_HB3(7));
f_priors+=neg_log_prior(log_q_cH,set_log_q_cH(5),set_log_q_cH(6),set_log_q_cH(7));
f_priors+=neg_log_prior(log_q_HB,set_log_q_HB(5),set_log_q_HB(6),set_log_q_HB(7));
f_priors+=neg_log_prior(log_q_GR,set_log_q_GR(5),set_log_q_GR(6),set_log_q_GR(7));
f_priors+=neg_log_prior(F_init,set_F_init(5),set_F_init(6),set_F_init(7));
FUNCTION dvariable SR_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const dvariable& SSB, int func)
//Spawner-recruit function (Beverton-Holt or Ricker)
//-----------------------------------------------------------------------------------
//Double Gaussian function: 6 parameters (as in SS3)
//-----------------------------------------------------------------------------------
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)
//Jointed logistic function: 6 parameters (increasing and decreasing logistics joined at peak selectivity)
 //-----------------------------------------------------------------------------------
FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
//Logistic function: 4 parameters
//-----------------------------------------------------------------------------------
FUNCTION dvar_vector logistic_exponential(const dvar_vector& ages, const dvariable& L50, const dvariable& slope, const dvariable& sigma, const dvariable& joint)
//Logistic-exponential: 4 parameters (but 1 is fixed)
//-----------------------------------------------------------------------------------
FUNCTION dvar_vector logistic(const dvar_vector& ages, const dvariable& L50, const dvariable& slope)
//Logistic function: 2 parameters
//-----------------------------------------------------------------------------------
FUNCTION logistic(exponential(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)
//Logistic function: 6 parameters (as in SS3)

FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
//Logistic function: 4 parameters (but 1 is fixed)

FUNCTION dvar_vector logistic_joint(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
//Logistic function: 6 parameters (as in SS3)

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

FUNCTION logistic(exponential(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)
//Logistic function: 6 parameters (as in SS3)

FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
//Logistic function: 4 parameters (but 1 is fixed)

FUNCTION dvar_vector logistic_joint(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2)
//Logistic function: 6 parameters (as in SS3)

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

FUNCTION logistic(exponential(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1, const dvariable& L502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)
//Logistic function: 6 parameters (as in SS3)
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;

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
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;

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
RETURN_ARRAYS_INCREMENT();

dvariable EffN_Tmp; dvariable numer; dvariable denom;
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;

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
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;

FUNCTION dvariable lk_robust_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;
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;
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;

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), pdf=prior type (1=none, 2=lognormal, 3=normal, 4=beta)
RETURN_ARRAYS_INCREMENT();

dvariable LkvalTmp;
dvariable alpha, beta, ab_iq;
dvariable big_number=1e10;
LkvalTmp=0.0;
for (int i=1; i<i<ncomp; i++)
{if (nsamp(i)<minSS)
{LkvalTmp+=-log((-log(Eprime(ii)+log(small_number)+mfexp(elem_div((-square(obs_comp(ii)-pred_comp(ii))) , (Eprime(ii)*0.0/nsamp_wgt(ii)) ))) ));
}
RETURN_ARRAYS_DECREMENT();
return LkvalTmp;
// 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(var<0.0) LkvalTmp=big_number=1e10; // convert cv to variance on log scale
            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);
        break;
}
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)
    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)
    RETURN_ARRAYS_INCREMENT();
    dvariable SdnrTmp;
    dvariable small_number=0.00001;
    dvariable n;
    dvar_vector res(cv.indexmin(),cv.indexmax());
    SdnrTmp=0.0;
    for(int i=1; i<ncomp; i++)
    {
        n=(sum(elem_prod(square(ages),pred_comp(i))));
        res(i)=sqrt(sum(elem_prod(square(ages),pred_comp(i))));
        SdnrTmp+=sum(elem_div((res-p(i)),sqrt(res(i))/ncomp));
    }
    RETURN_ARRAYS_DECREMENT();
    return SdnrTmp;

//-----------------------------------------------------------------------------------
//REPORT SECTION
if (last_phase())
{
    cout<<"start report"<<endl;
    get_weighted_current();
    cout<<"got weighted"<<endl;
    get_any();
    cout<<"get any"<<endl;
    get_redundant_stuff();
    cout<<"get m"<<endl;
    get_p=recruitstuff();
    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, ";
    cout<<minute<<" minutes, ";
    cout<<second<<" seconds"<<endl;
    cout<<"--TotalLikelihood: "<<fval<<""<<endl;
    cout<<"--Final gradient: "<<objective_function_value::pobjfun->gmax<<endl;
    cout<<endl;
cout << endl;
cout << "><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--><--}<