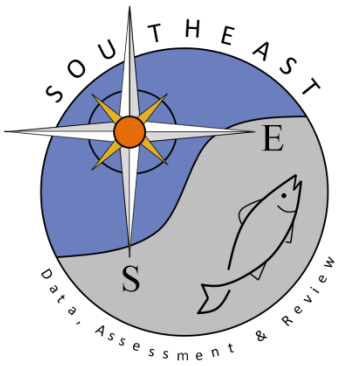
Model Diagnostics and Source Code for SEDAR 41 Gray Triggerfish (*Balistes capriscus*) Benchmark Stock Assessment

Sustainable Fisheries Branch, National Marine Fisheries Service
(Contact: Kevin Craig)

SEDAR41-RW05

Submitted: 1 March 2016



Please cite this document as:

SFB-NMFS. 2016. Model Diagnostics and Source Code for SEDAR 41 Gray Triggerfish (*Balistes capriscus*) Benchmark Stock Assessment. SEDAR41-RW05. SEDAR, North Charleston, SC. 195 pp.

Notice on SEDAR Working Papers

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

Model Diagnostics and Source Code for SEDAR 41 Gray Triggerfish (*Balistes capriscus*) Benchmark Stock Assessment

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

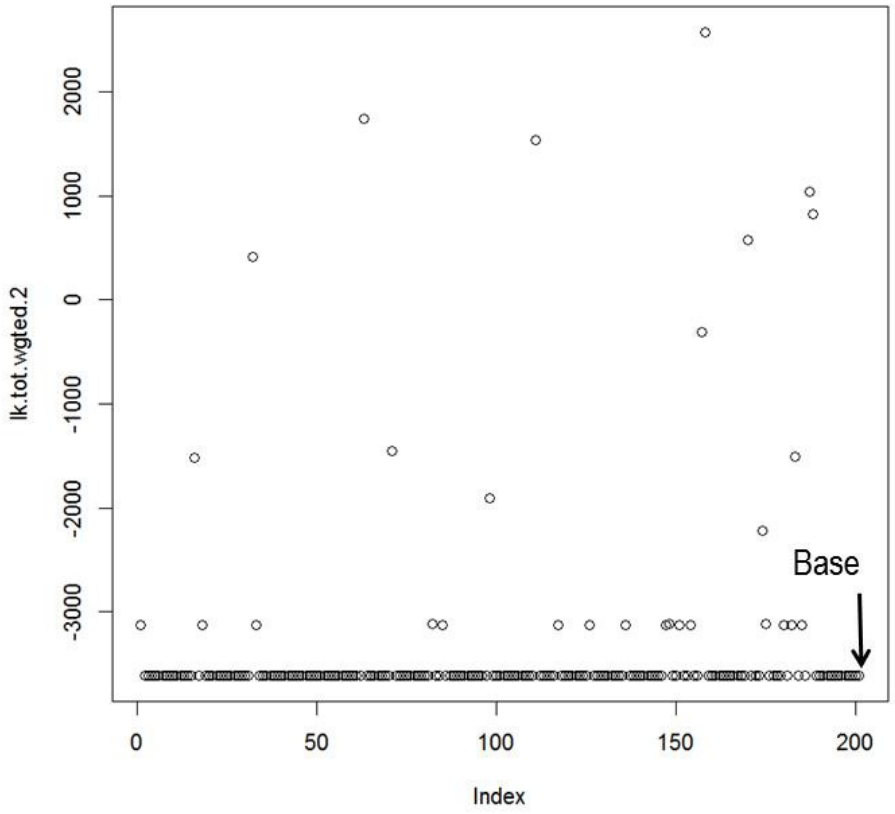
26-February-2016

Additional BAM diagnostics, analyses, and code

This working paper is a repository for supplementary materials generated during the SEDAR 41 Gray Triggerfish benchmark assessment. The plots are diagnostics or alternative visualizations of results presented in the assessment report. The source ADMB .tpl file and the .dat data file for the BAM catch-age model tailored to Gray Triggerfish follow the plots.

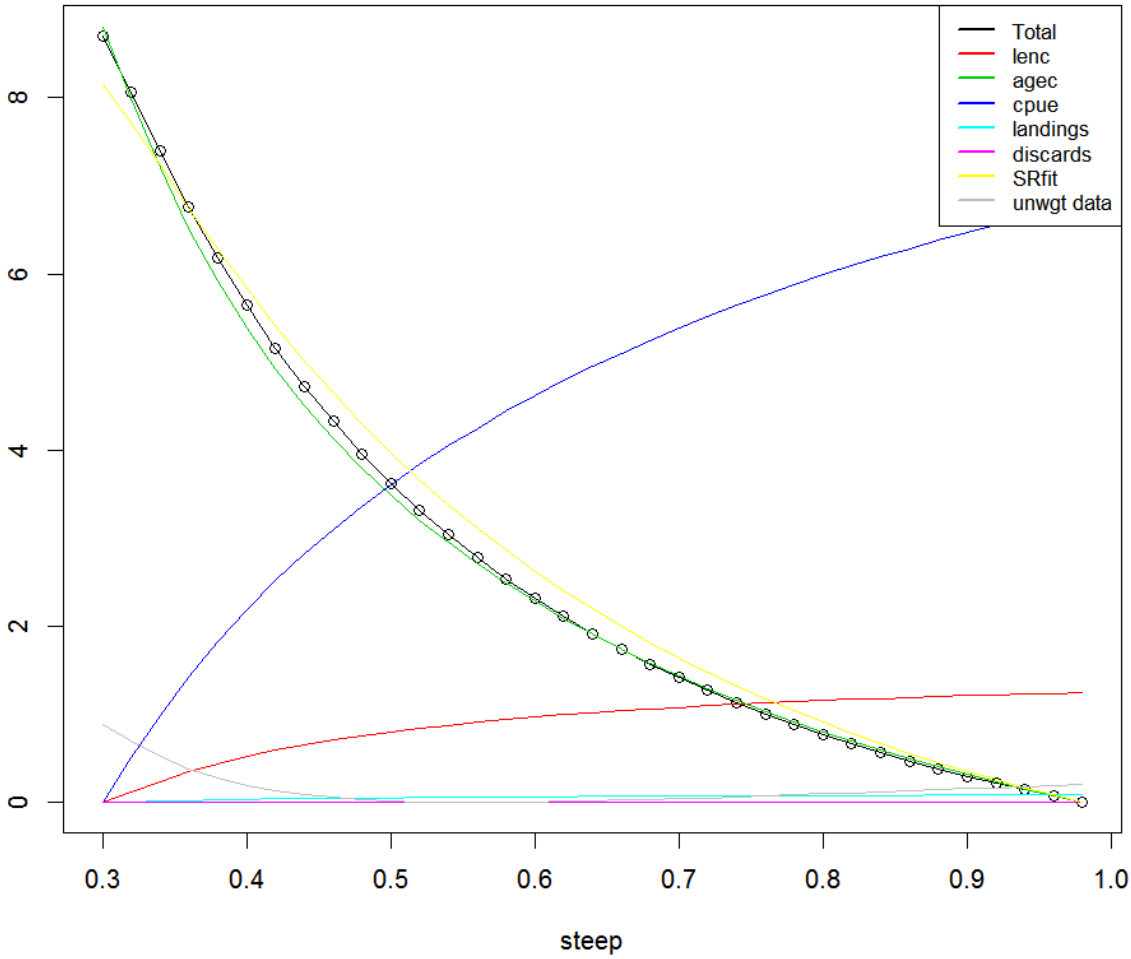
Starting Value Analysis:

This analysis was conducted to determine if results from the BAM catch-age model were sensitive to the starting values used for the estimated parameters. A new starting values for each estimated parameter was randomly drawn from a uniform distribution with a lower and upper bound set to ±25% of the original starting value. Estimated parameters included growth curve parameters (Linf, k, t0, and len_CV), recruitment parameters (R0 and rec_sigma), all selectivy parameters, and initial fishing mortality (F_init). The drawn value, the corresponding estimate, and the overall model likelihood were tabulated and compared at the end of 200 trials. For Gray Triggerfish, the base run (number 201 in the below plot) was at the global minimum.

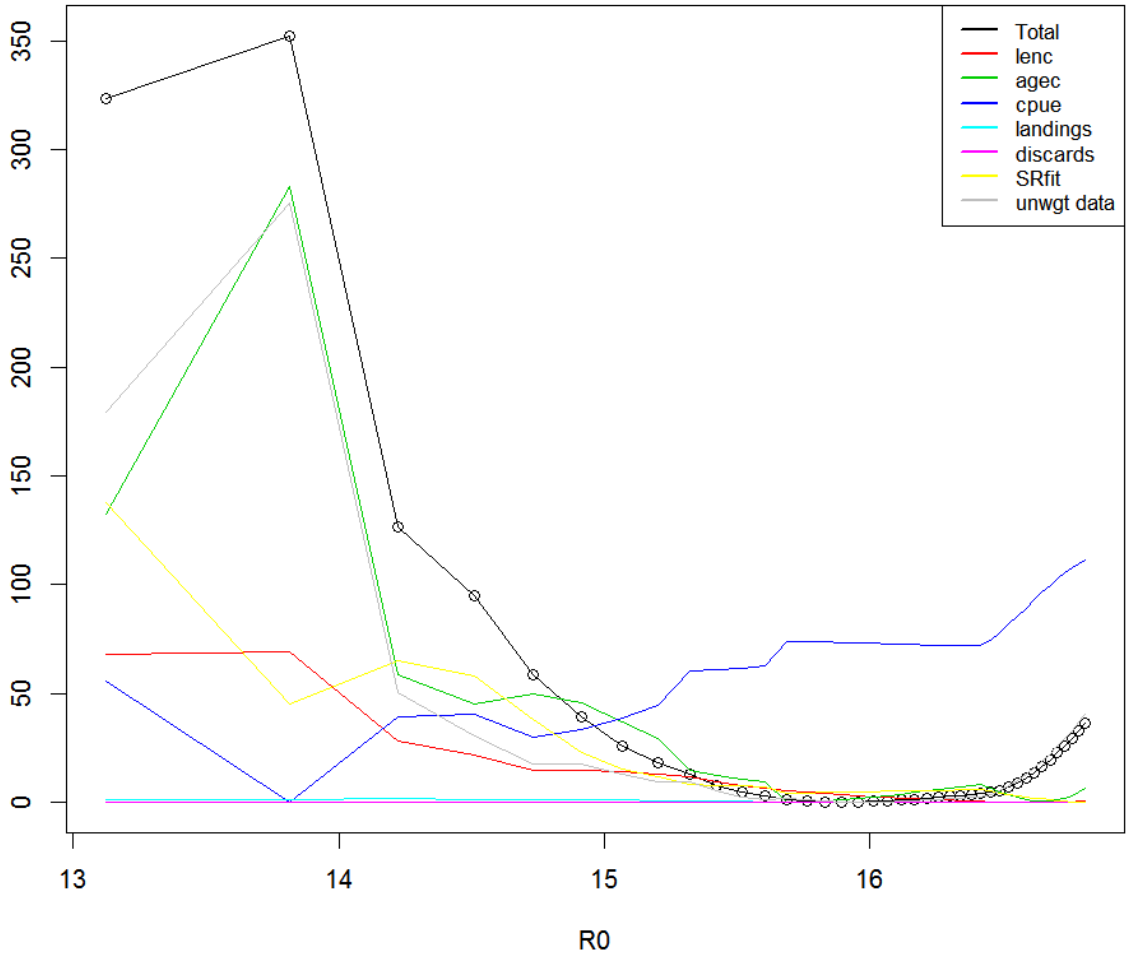


Likelihood profiles:

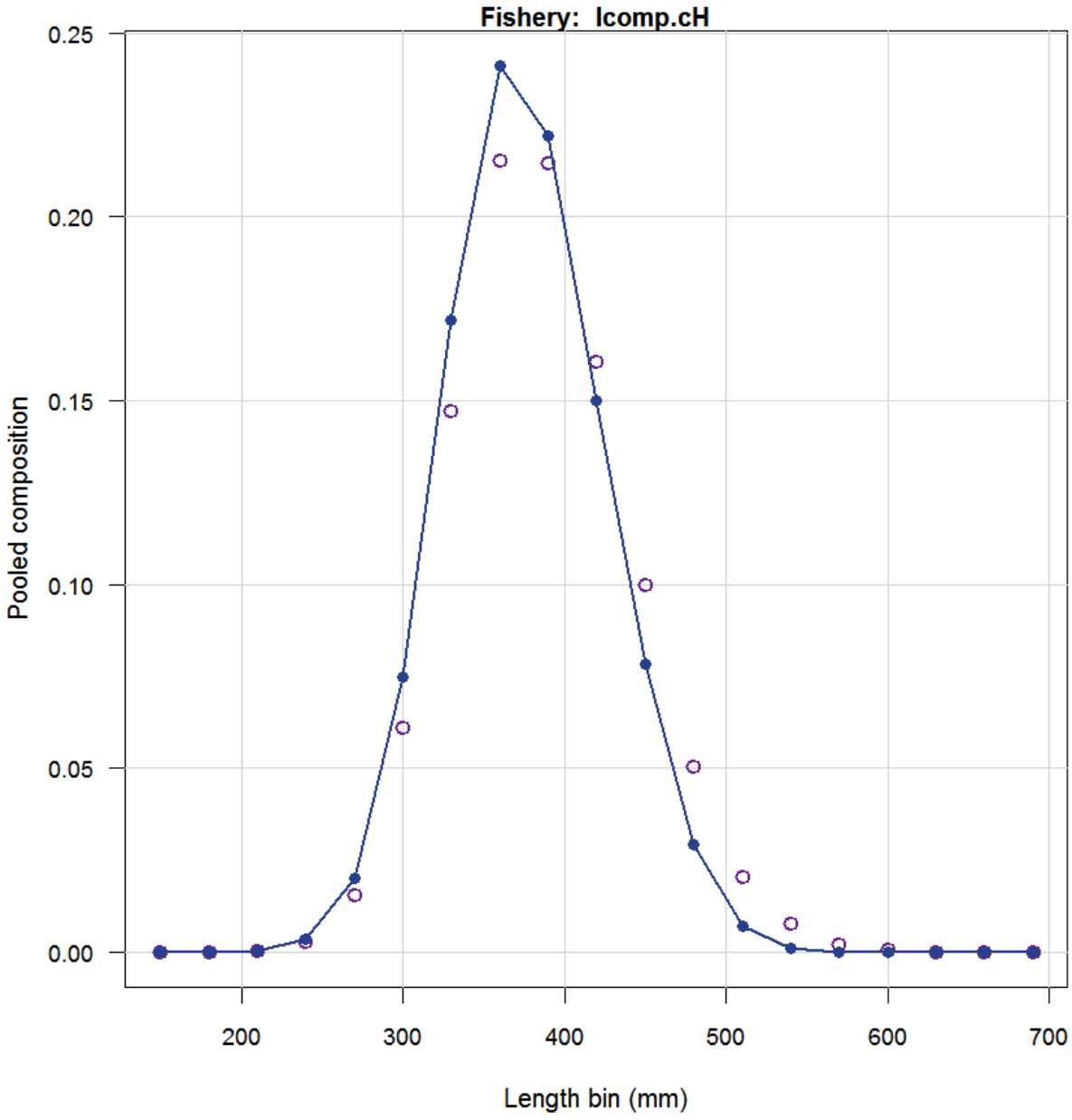
A range of steepness values were used over which to profile (h=0.3 to 0.98):

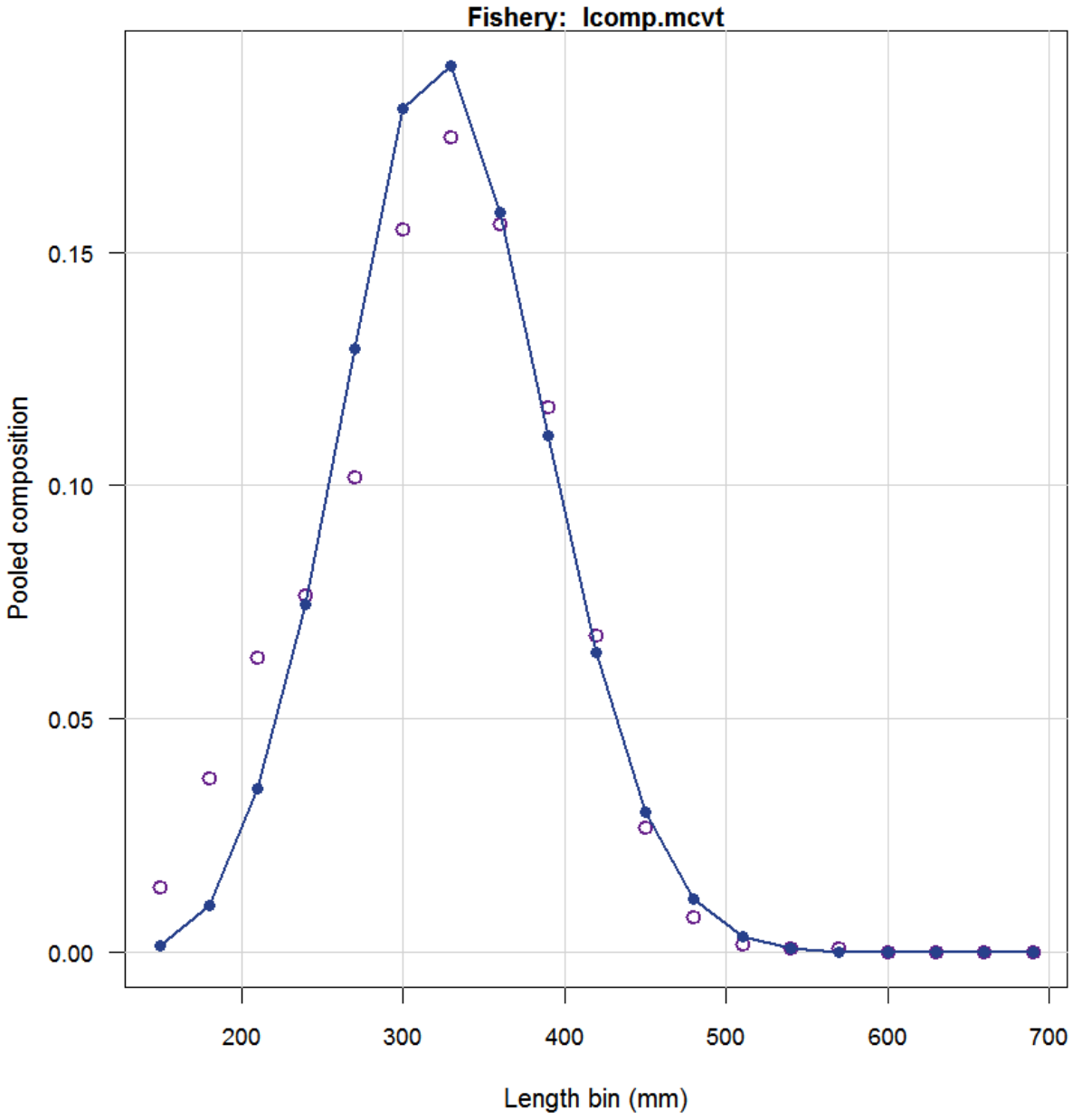


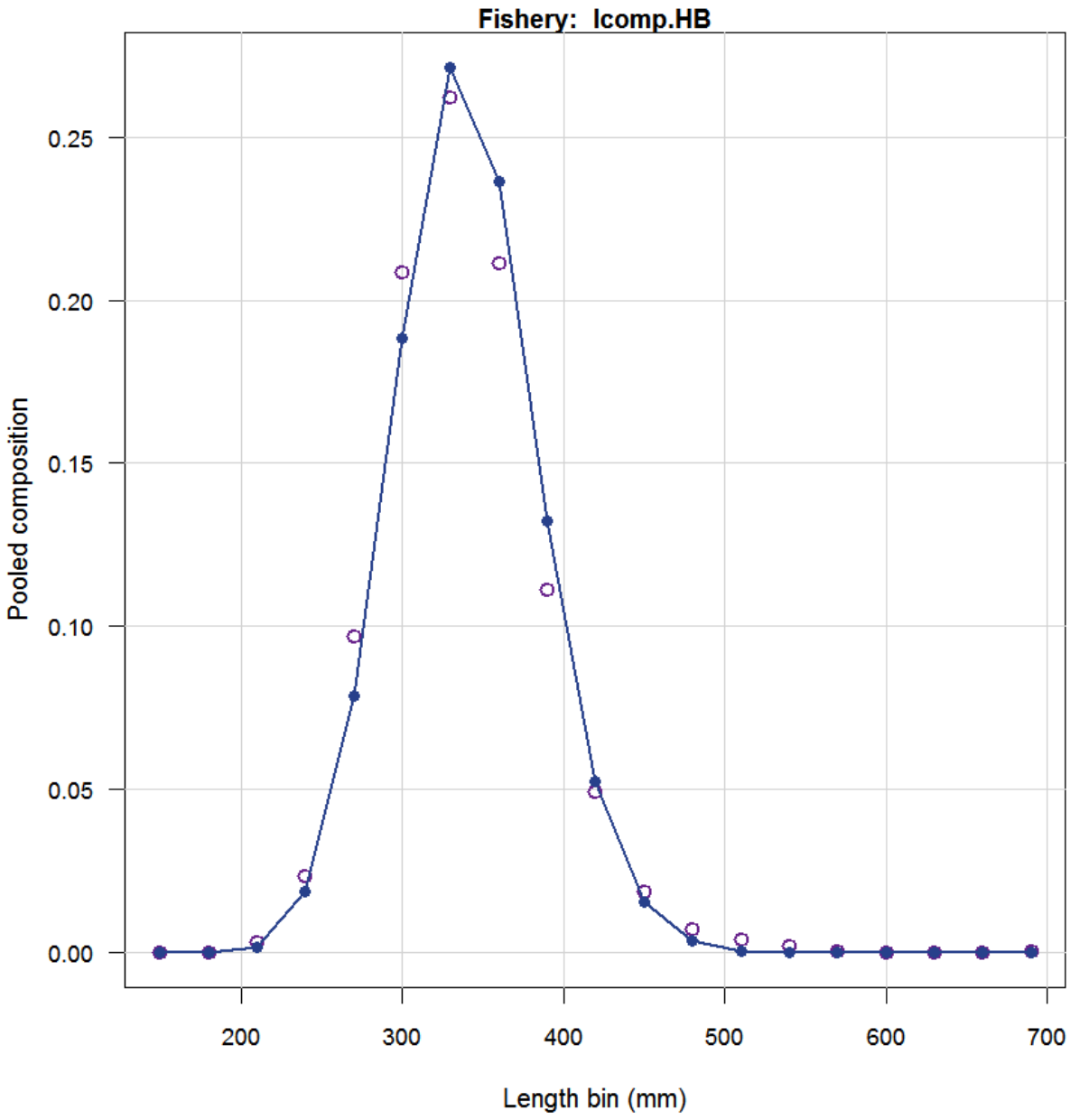
R0 was profiled using a range from 500,000 to 20,000,000 (13-17 on log scale):

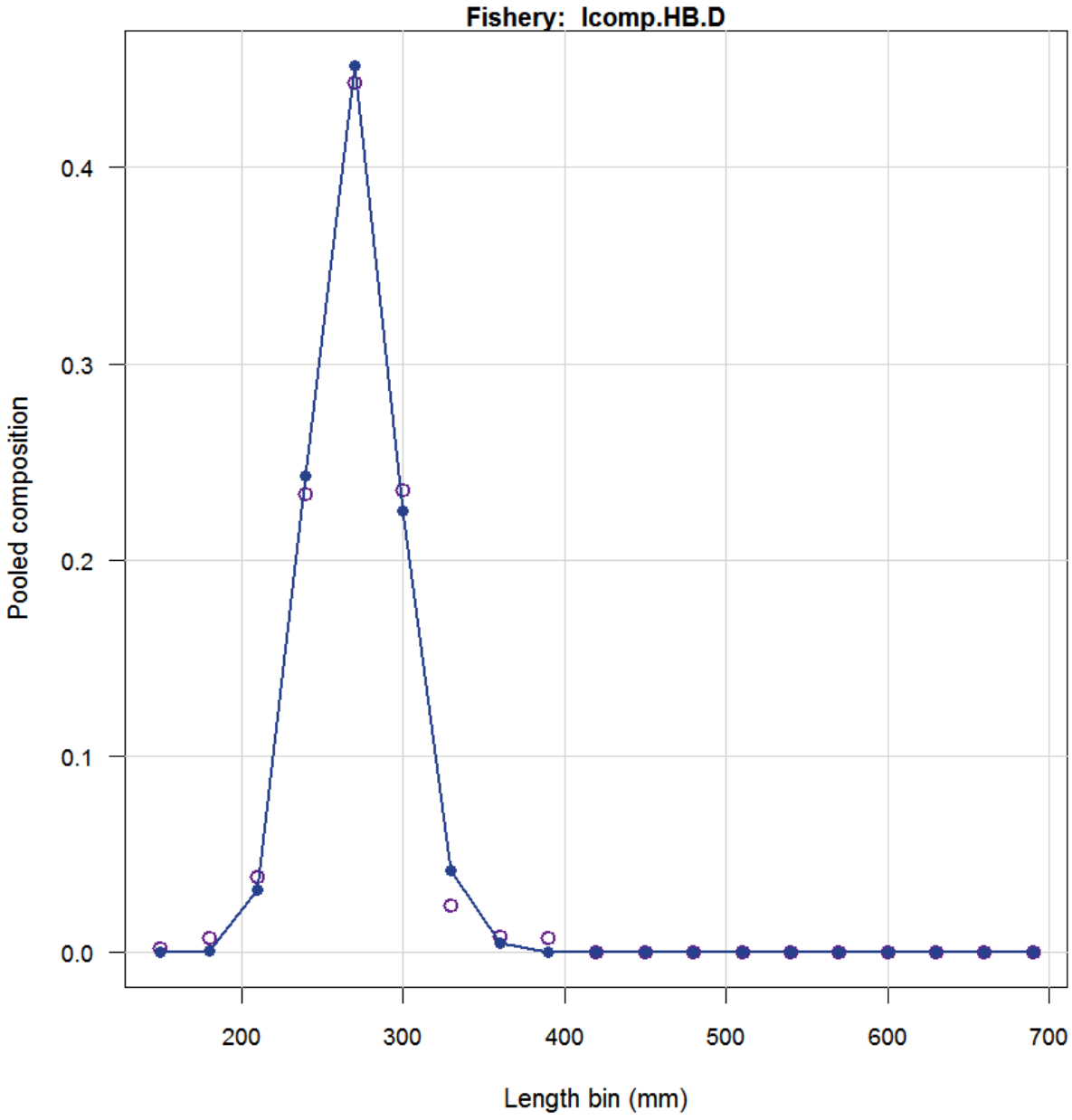


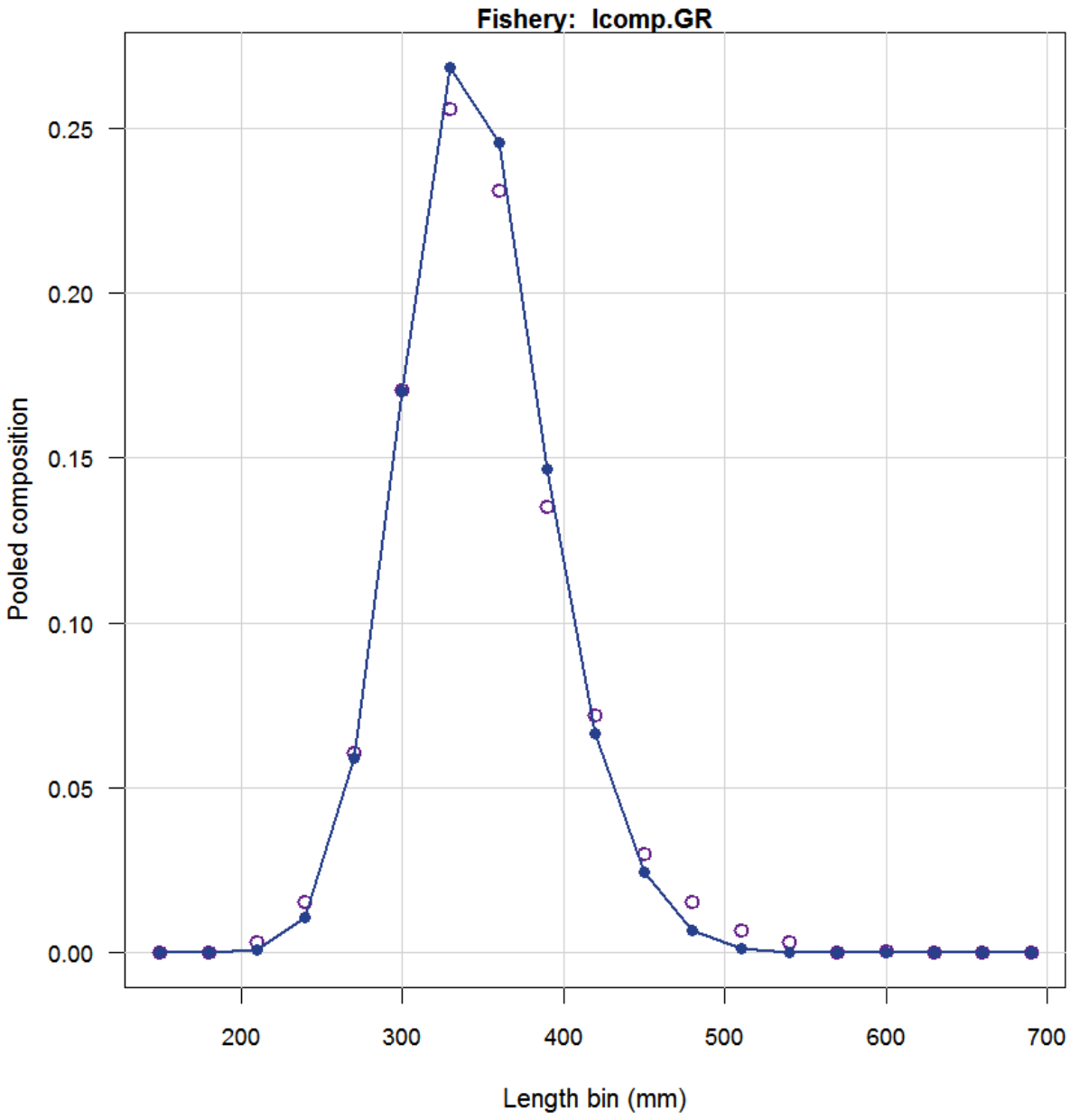
Diagnostic and supplementary plots:

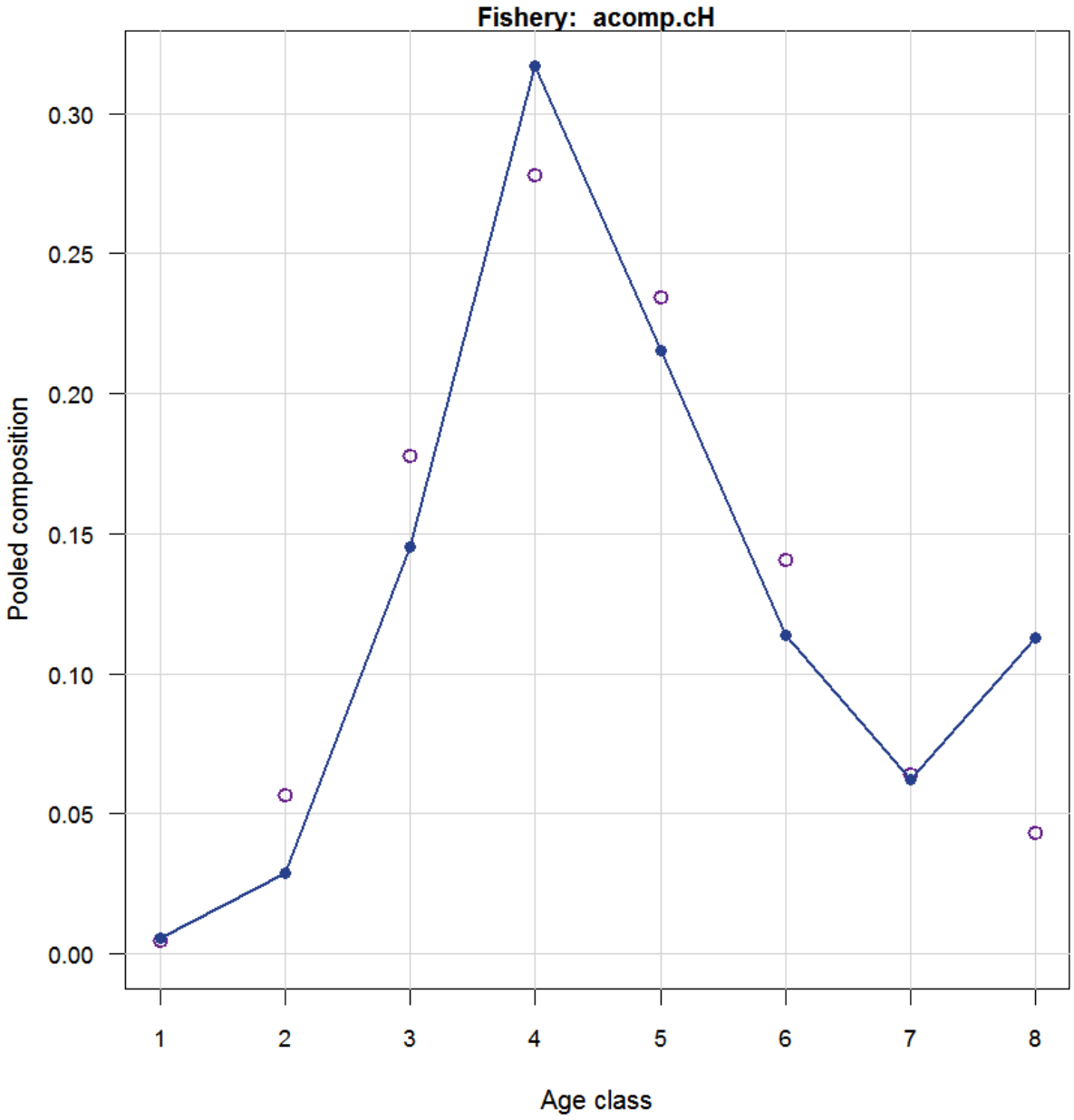


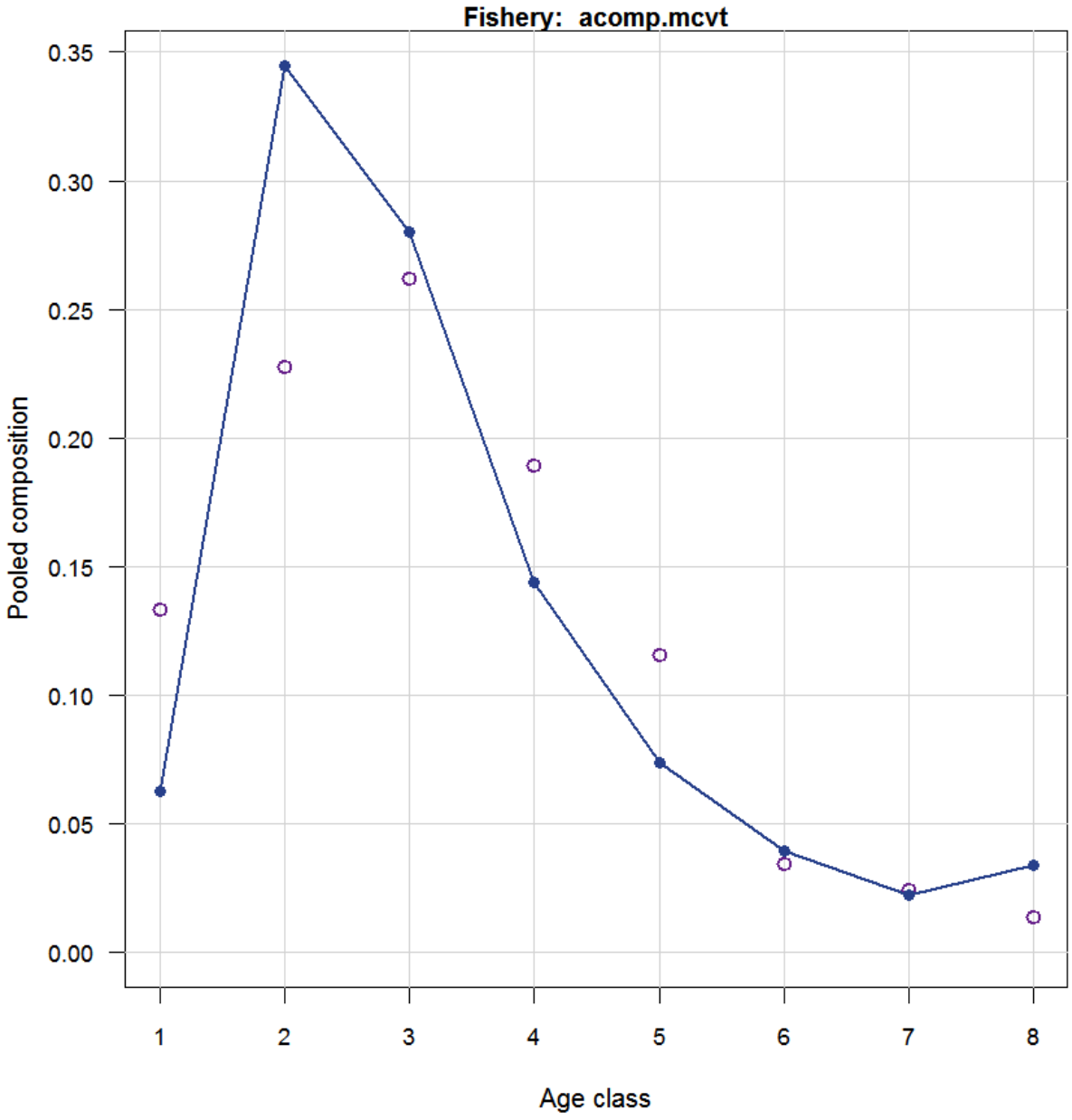

Fishery: lcomp.mcvt
Pooled composition
0.00
0.05
0.10
0.15
200
300
400
500
600
700
Length bin (mm)

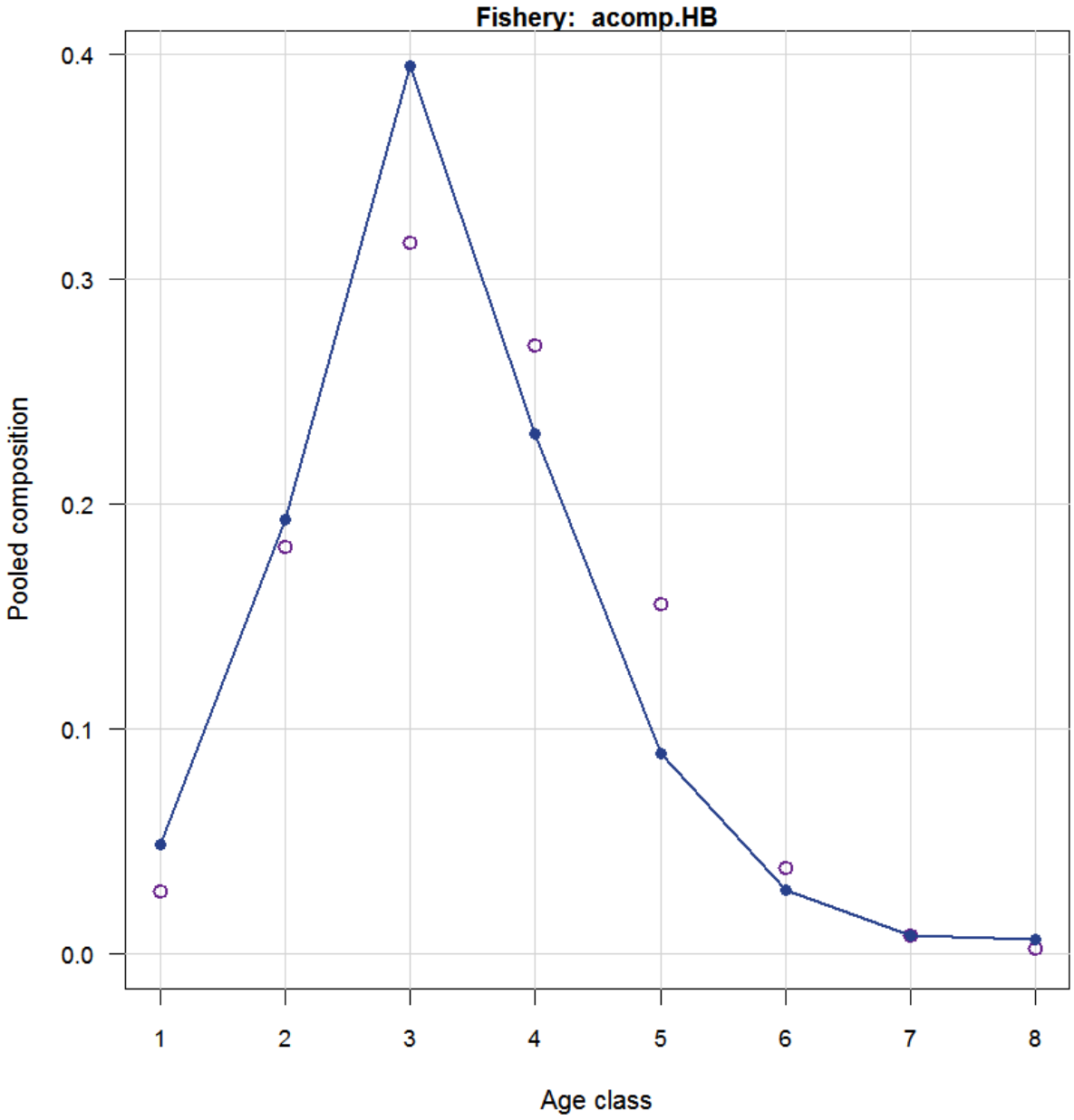

Fishery: lcomp.HB
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
200
300
400
500
600
700
Length bin (mm)

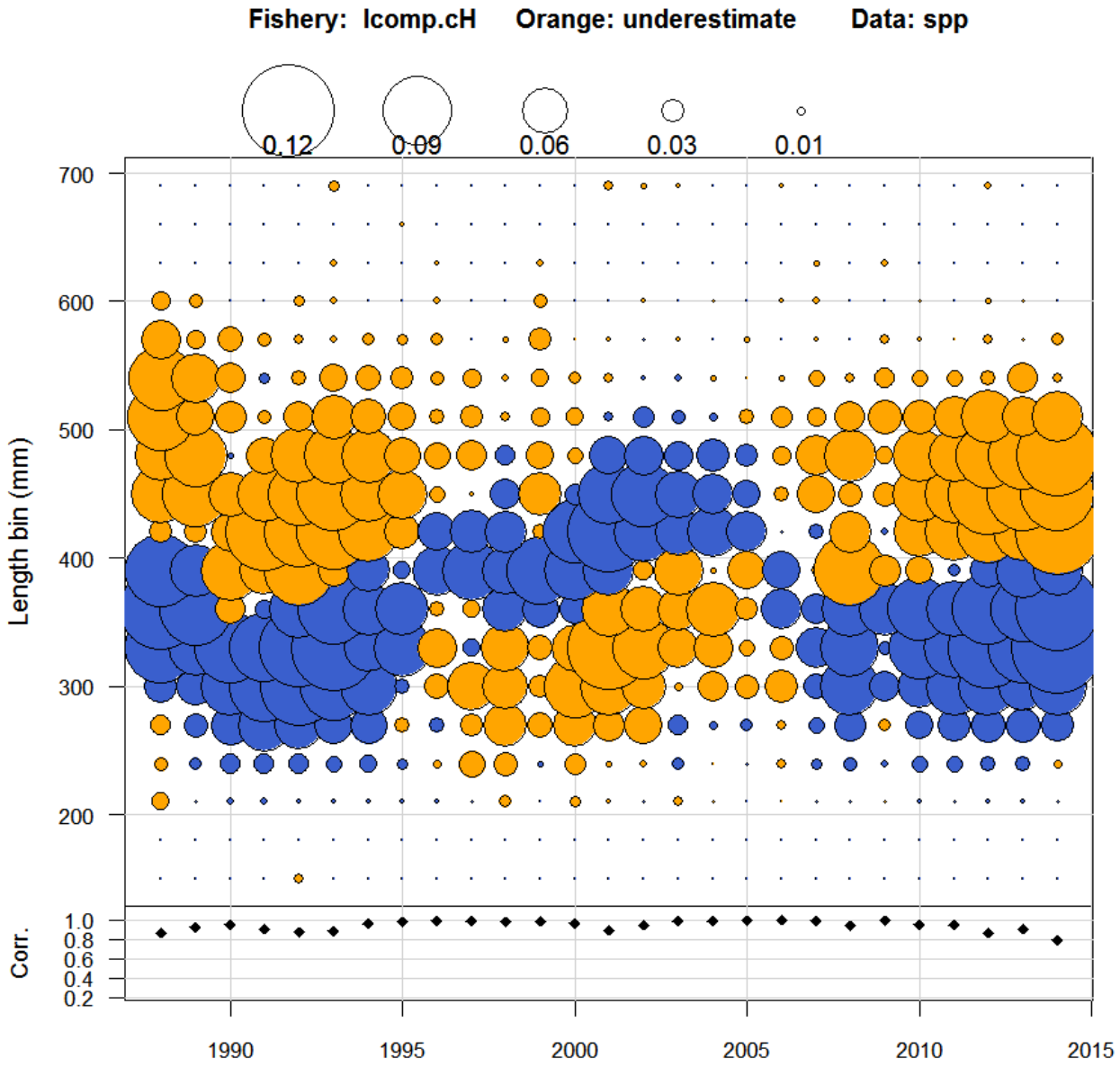

Fishery: lcomp.HB.D
Pooled composition
0.0
0.1
0.2
0.3
0.4
200
300
400
500
600
700
Length bin (mm)

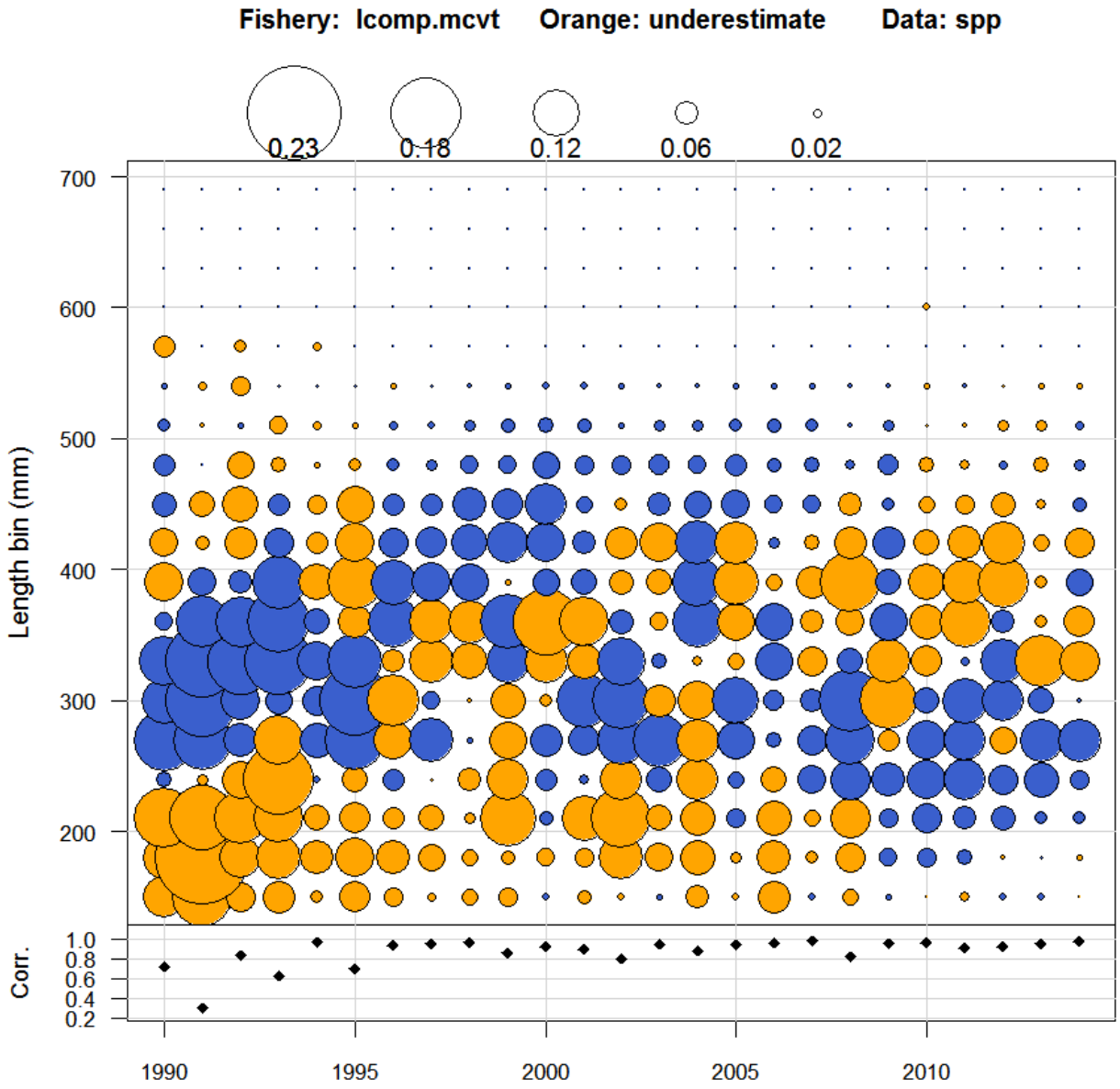

Fishery: lcomp.GR
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
200
300
400
500
600
700
Length bin (mm)

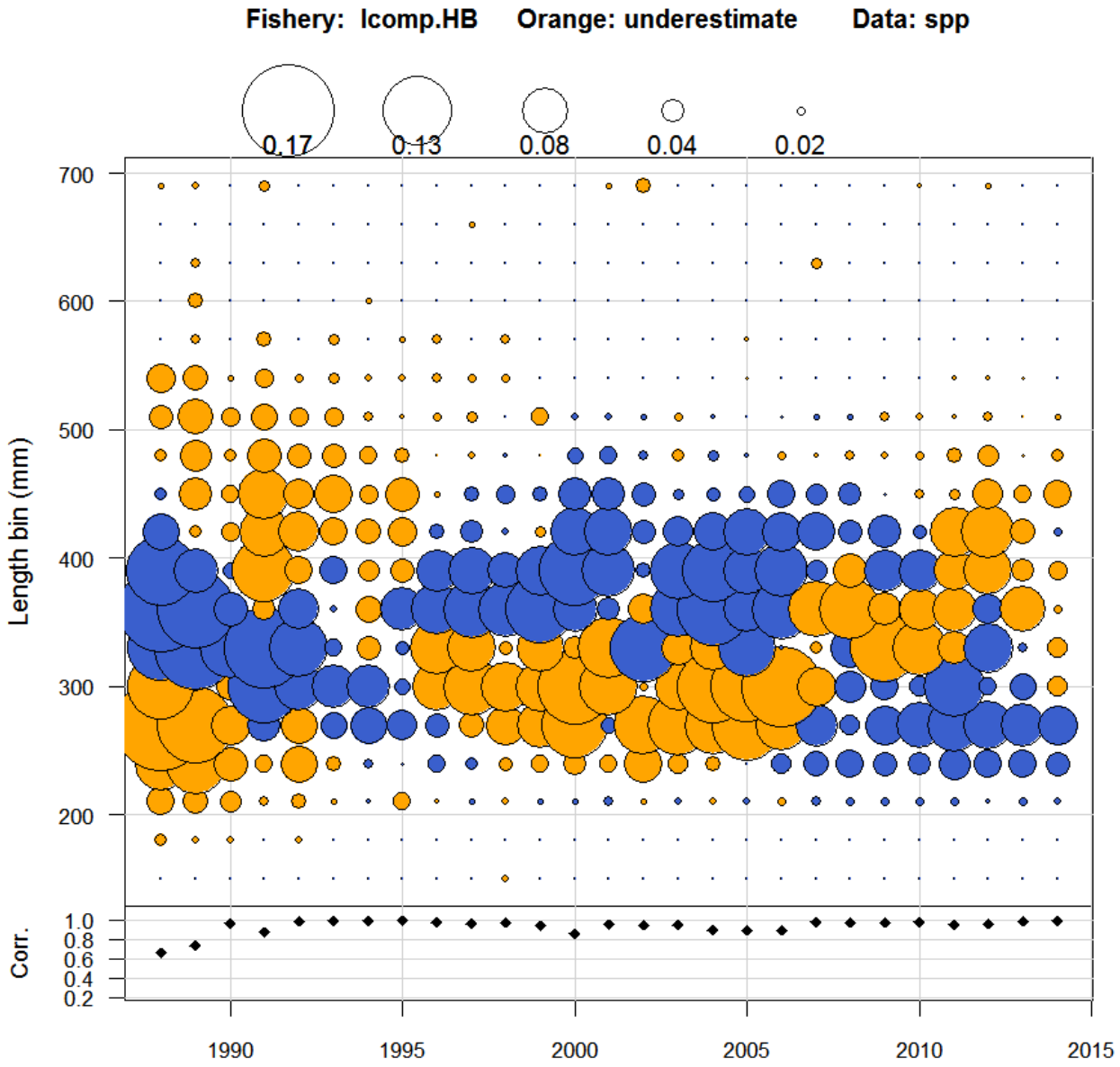

Fishery: acomp.cH
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
0.30
1
2
3
4
5
6
7
8
Age class

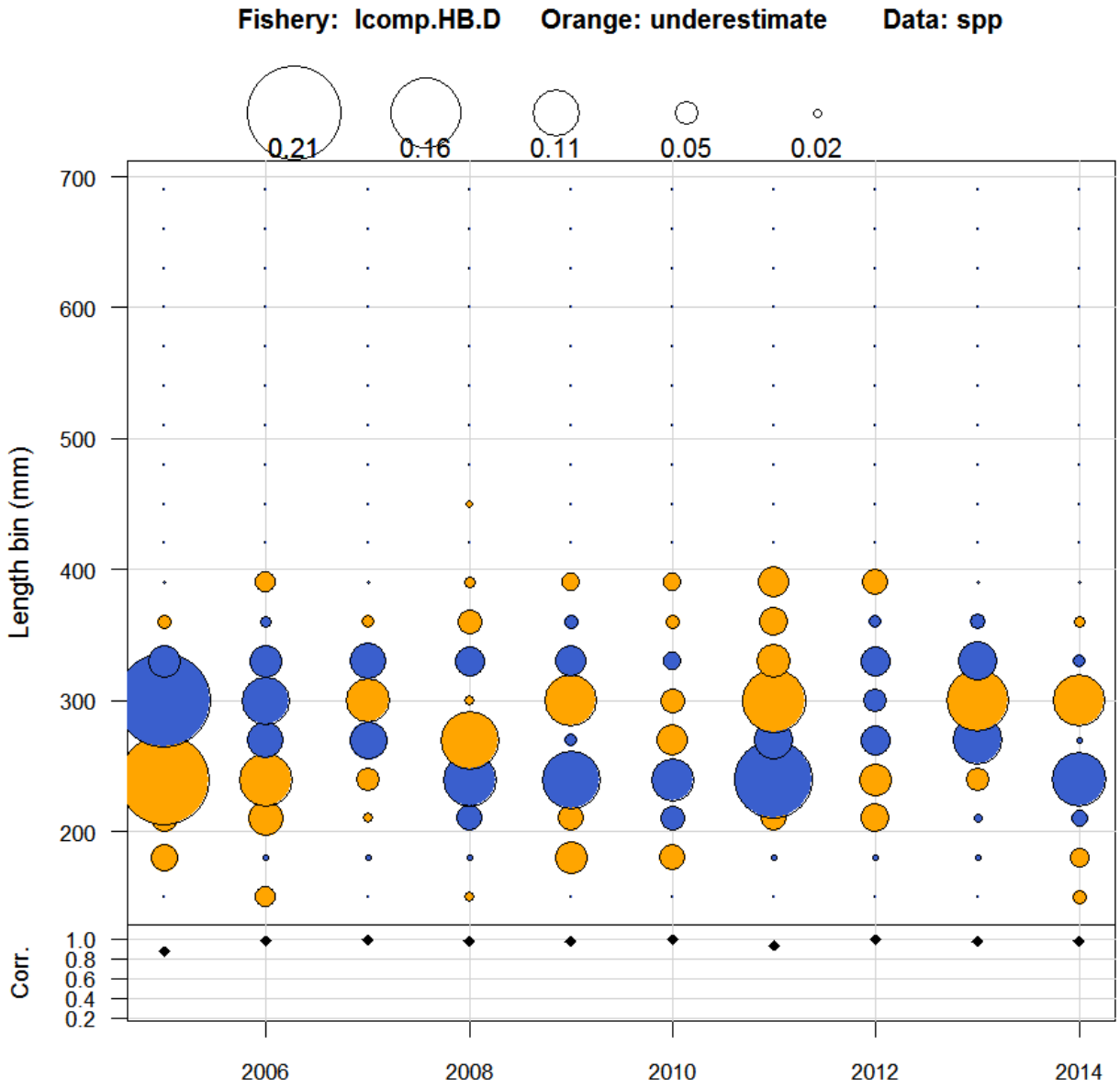

Fishery: acomp.mcvt
Pooled composition
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6
7
8
Age class

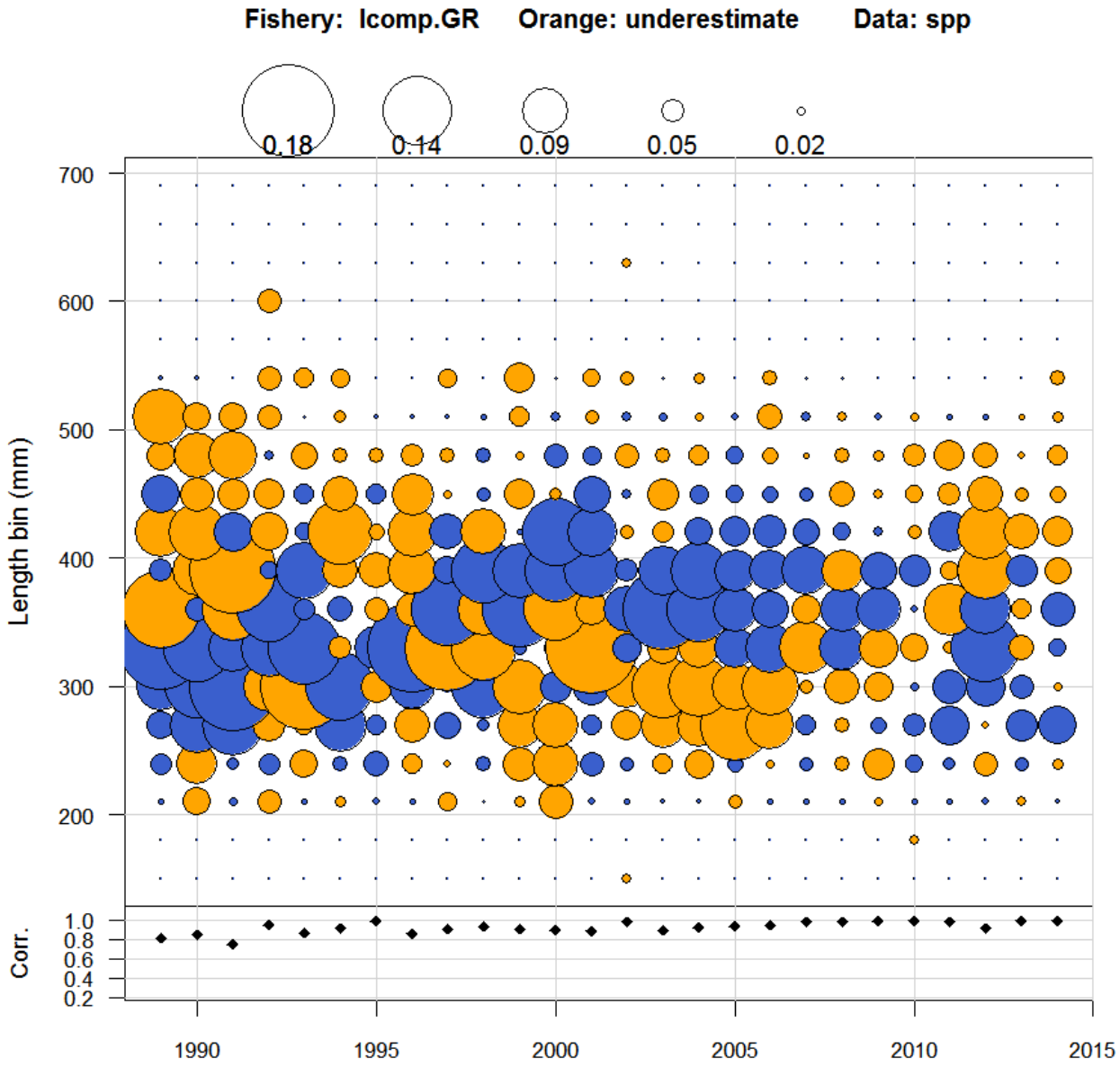

Fishery: acomp.HB
Pooled composition
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
8
Age class

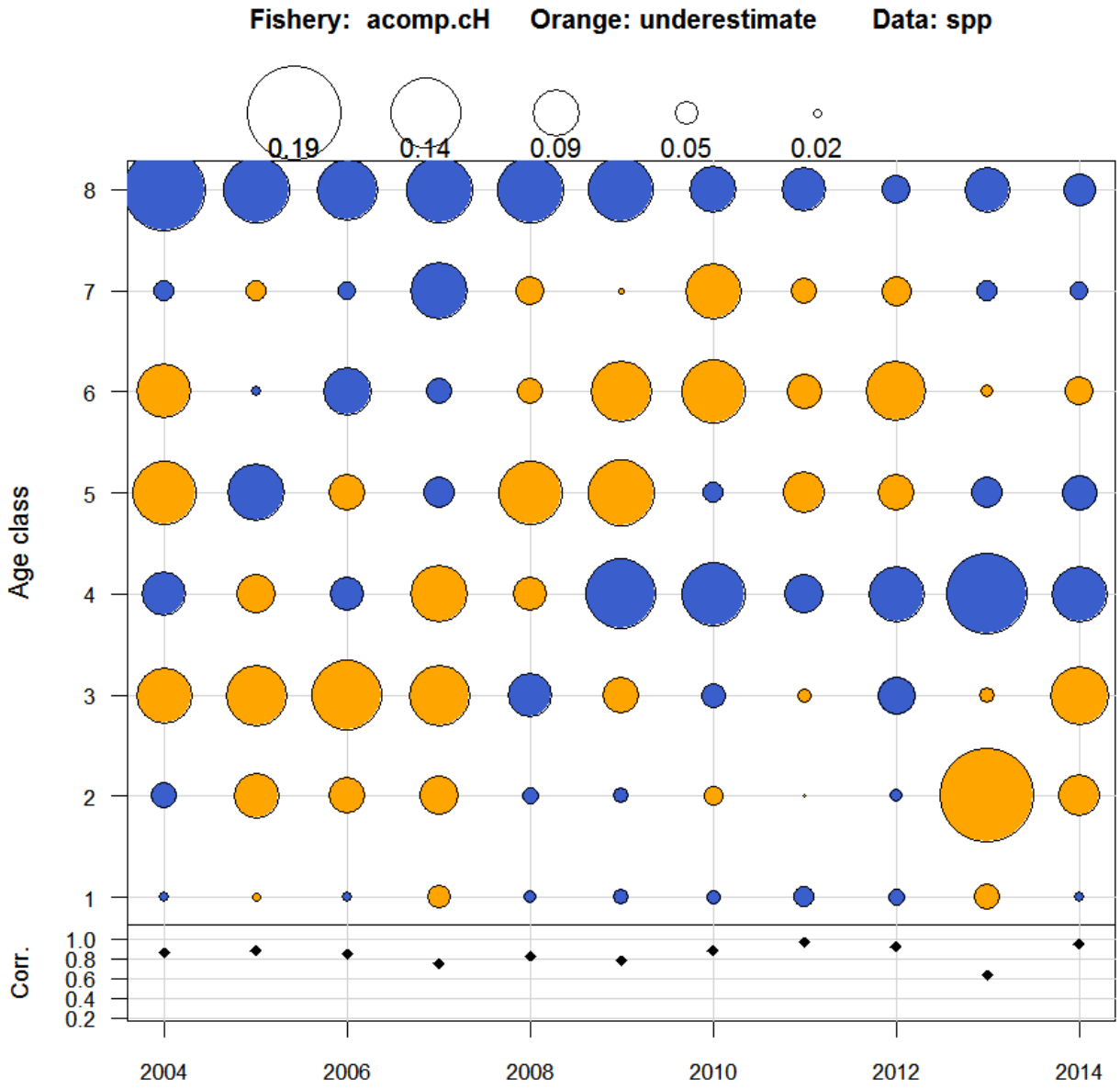

Fishery: lcomp.cH Orange: underestimate Data: spp
0.12
0.09
0.06
0.03
0.01
Length bin (mm)
700
600
500
400
300
200
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010
2015

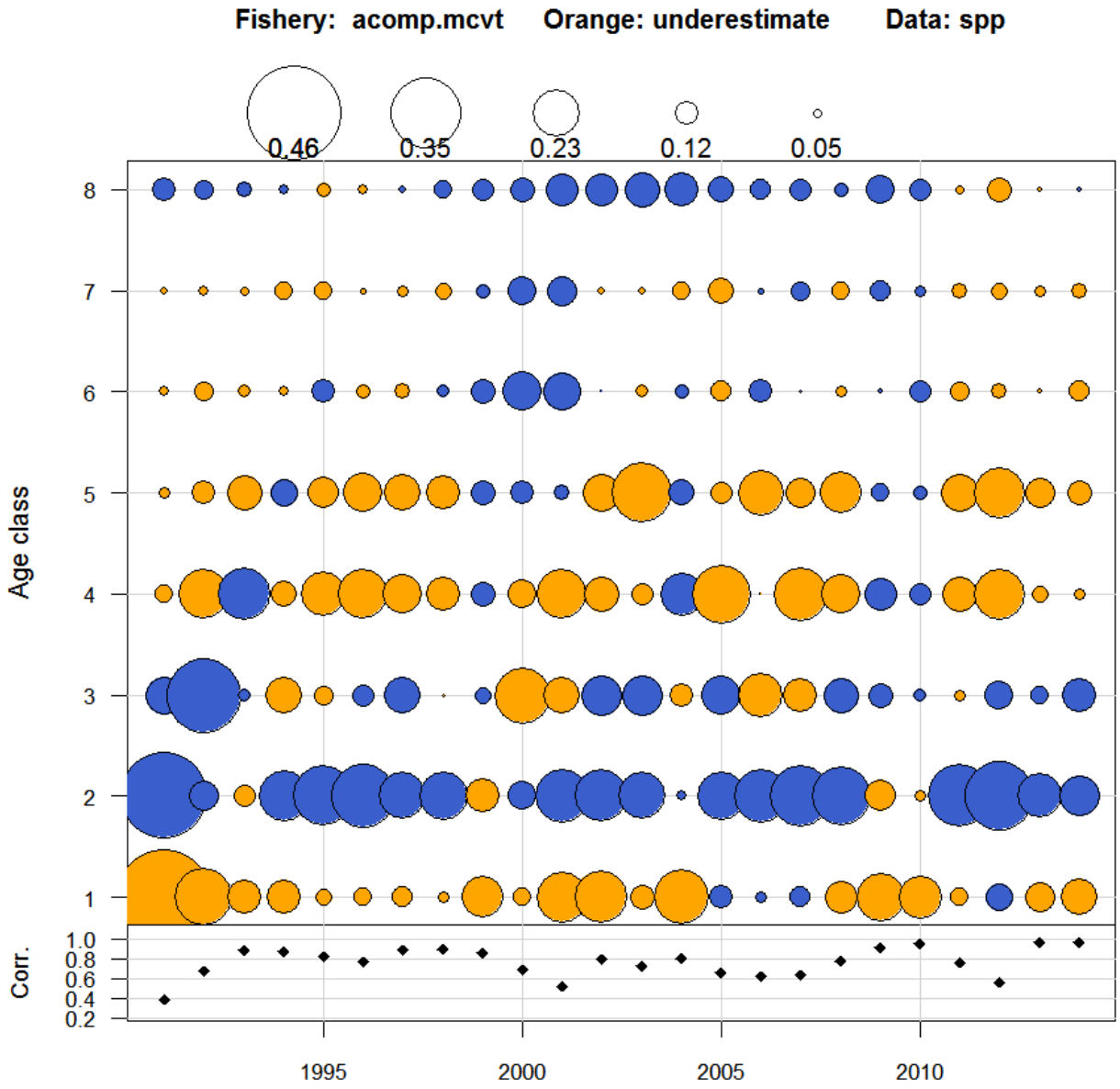

Fishery: lcomp.mcvt Orange: underestimate Data: spp
0.23
0.18
0.12
0.06
0.02
Length bin (mm)
700
600
500
400
300
200
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010

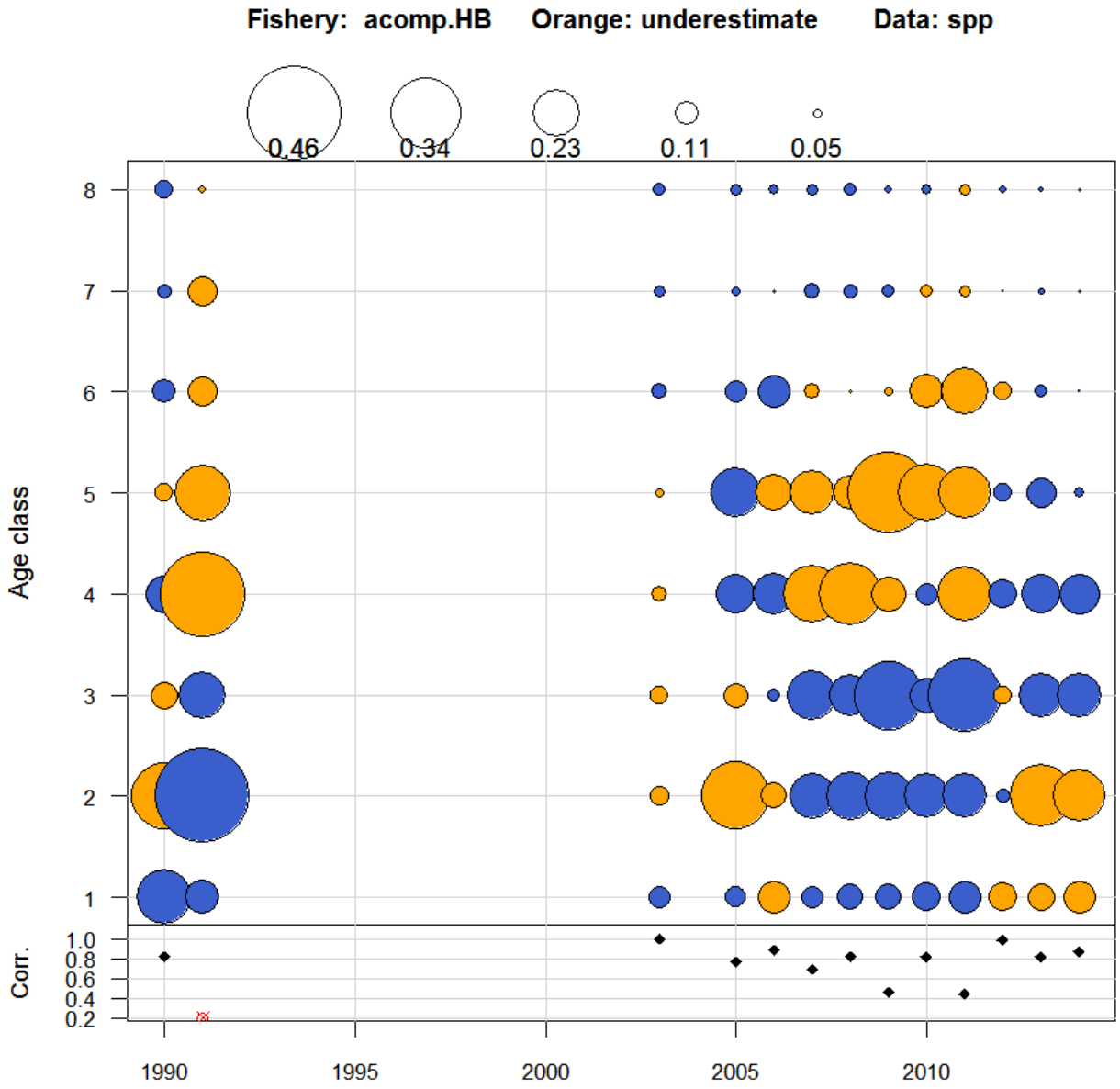

Fishery: lcomp.HB Orange: underestimate Data: spp
0.17
0.13
0.08
0.04
0.02
Length bin (mm)
700
600
500
400
300
200
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010
2015

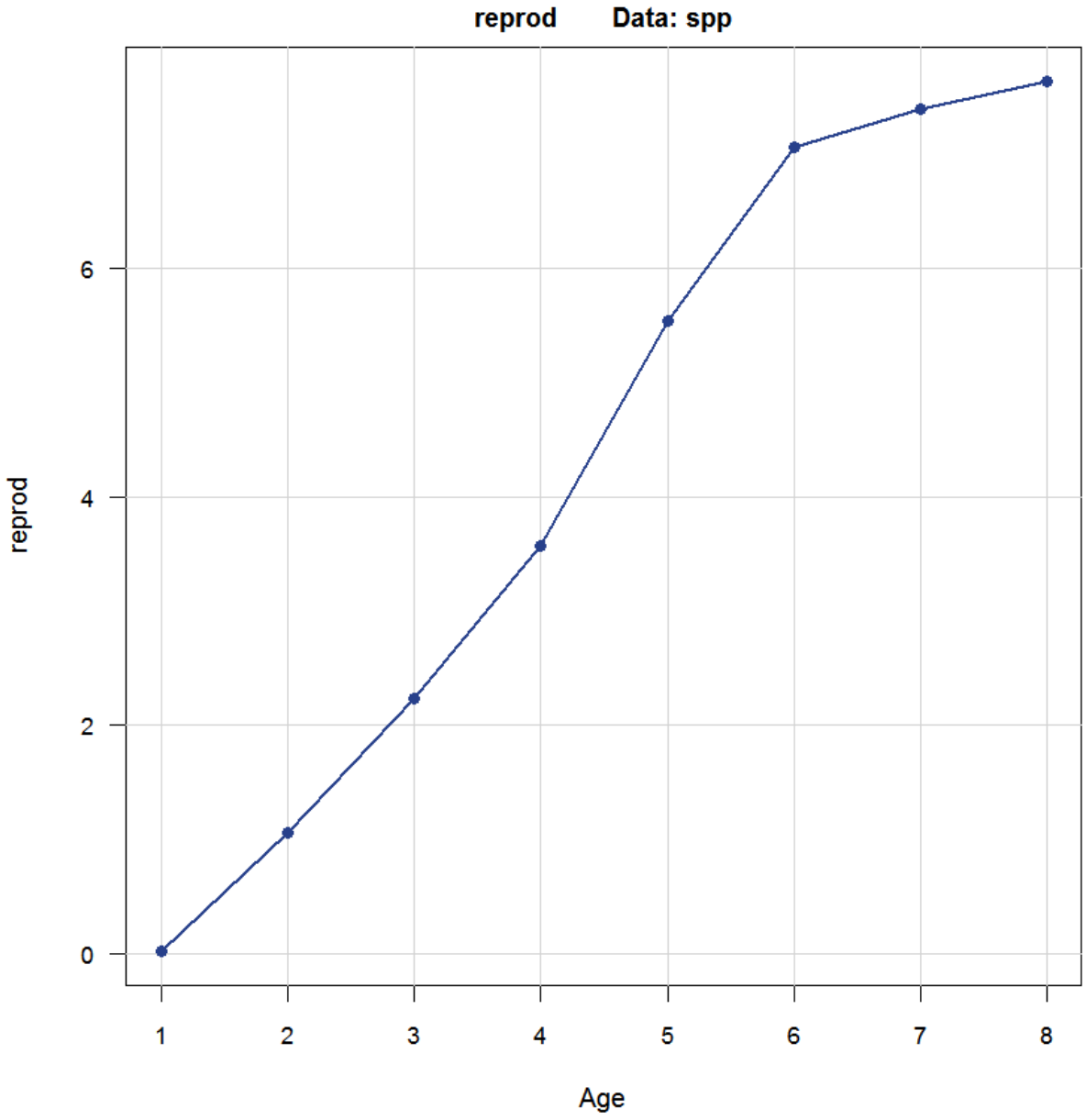

Fishery: lcomp.HB.D Orange: underestimate Data: spp
0.21
0.16
0.11
0.05
0.02
Length bin (mm)
700
600
500
400
300
200
Corr.
1.0
0.8
0.6
0.4
0.2
2006
2008
2010
2012
2014

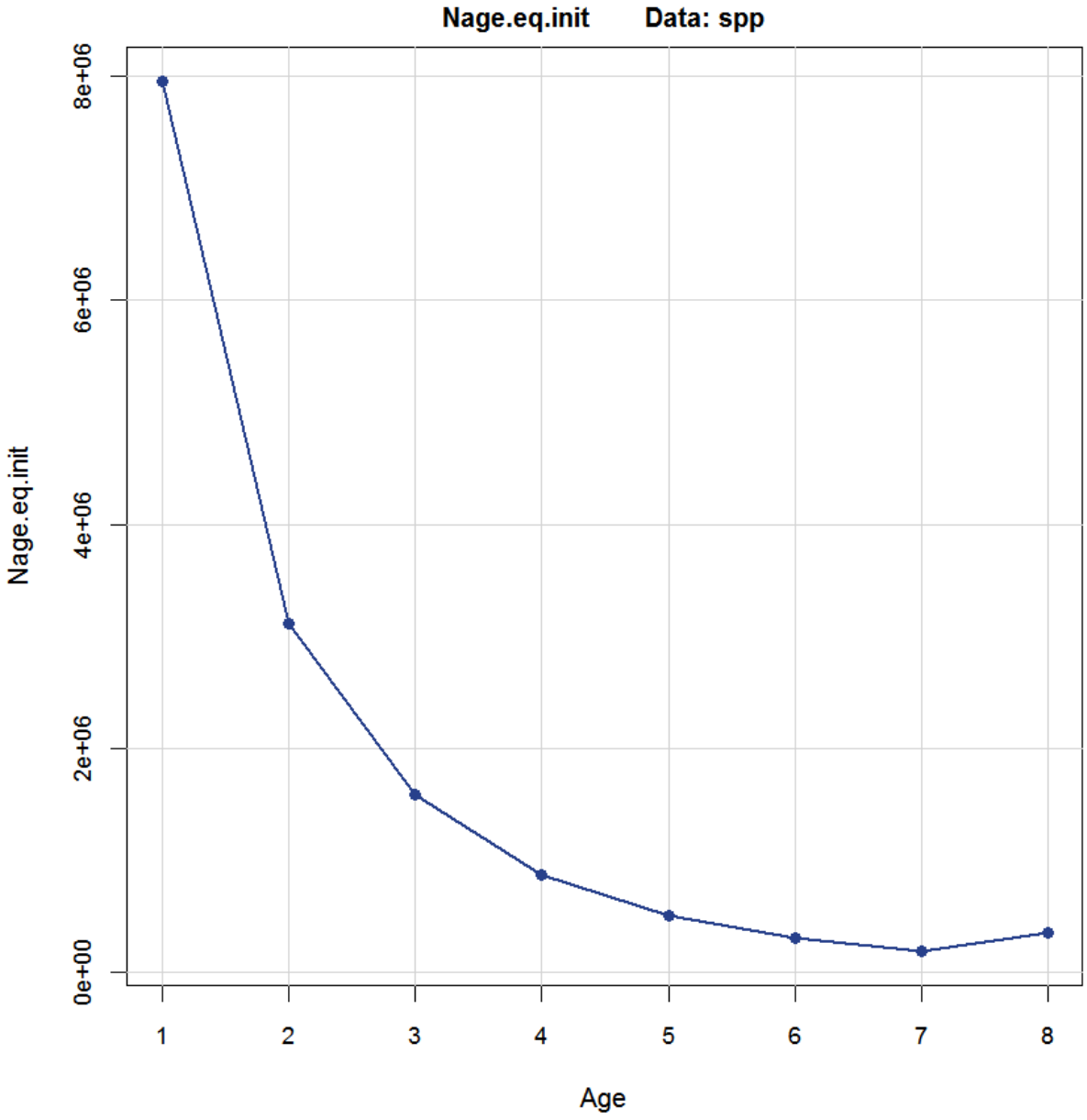

Fishery: lcomp.GR Orange: underestimate Data: spp
0.18
0.14
0.09
0.05
0.02
Length bin (mm)
700
600
500
400
300
200
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010
2015

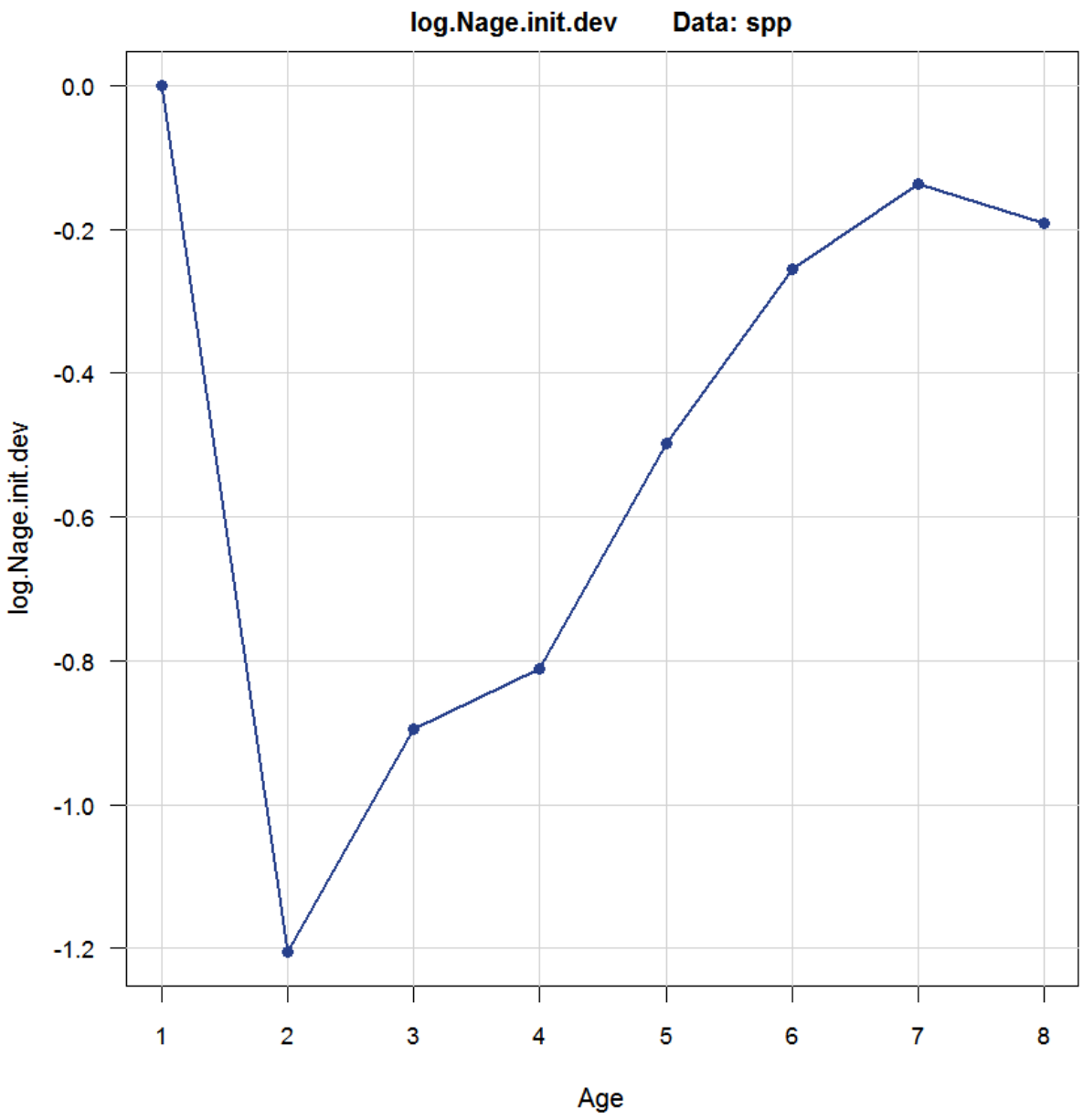

Fishery: acomp.cH Orange: underestimate Data: spp
0.19
0.14
0.09
0.05
0.02
Age class
8
7
6
5
4
3
2
1
Corr.
1.0
0.8
0.6
0.4
0.2
2004
2006
2008
2010
2012
2014

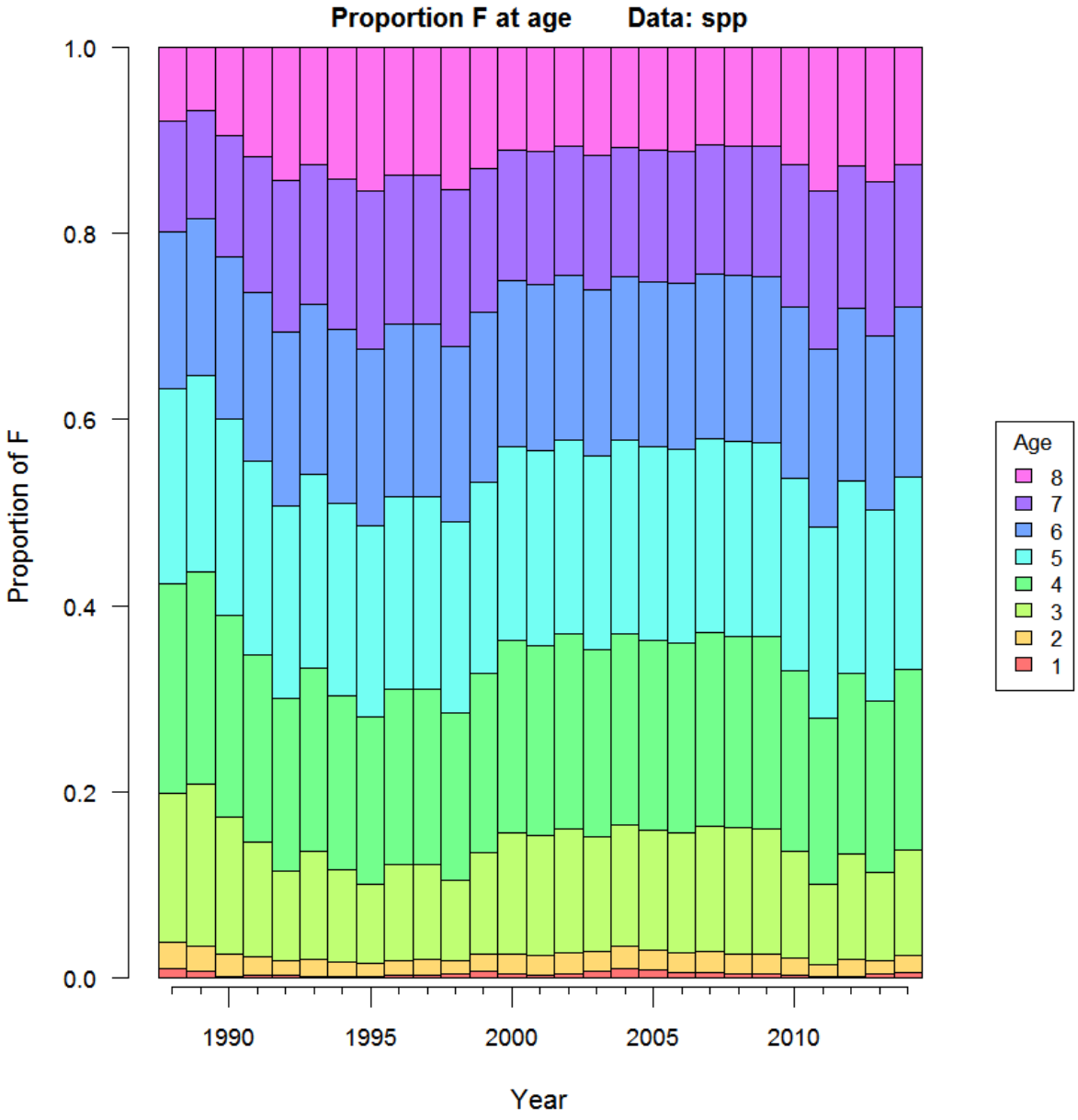

Fishery: acomp.mcvt Orange: underestimate Data: spp
0.46
0.35
0.23
0.12
0.05
Age class
8
7
6
5
4
3
2
1
Corr.
1.0
0.8
0.6
0.4
0.2
1995
2000
2005
2010

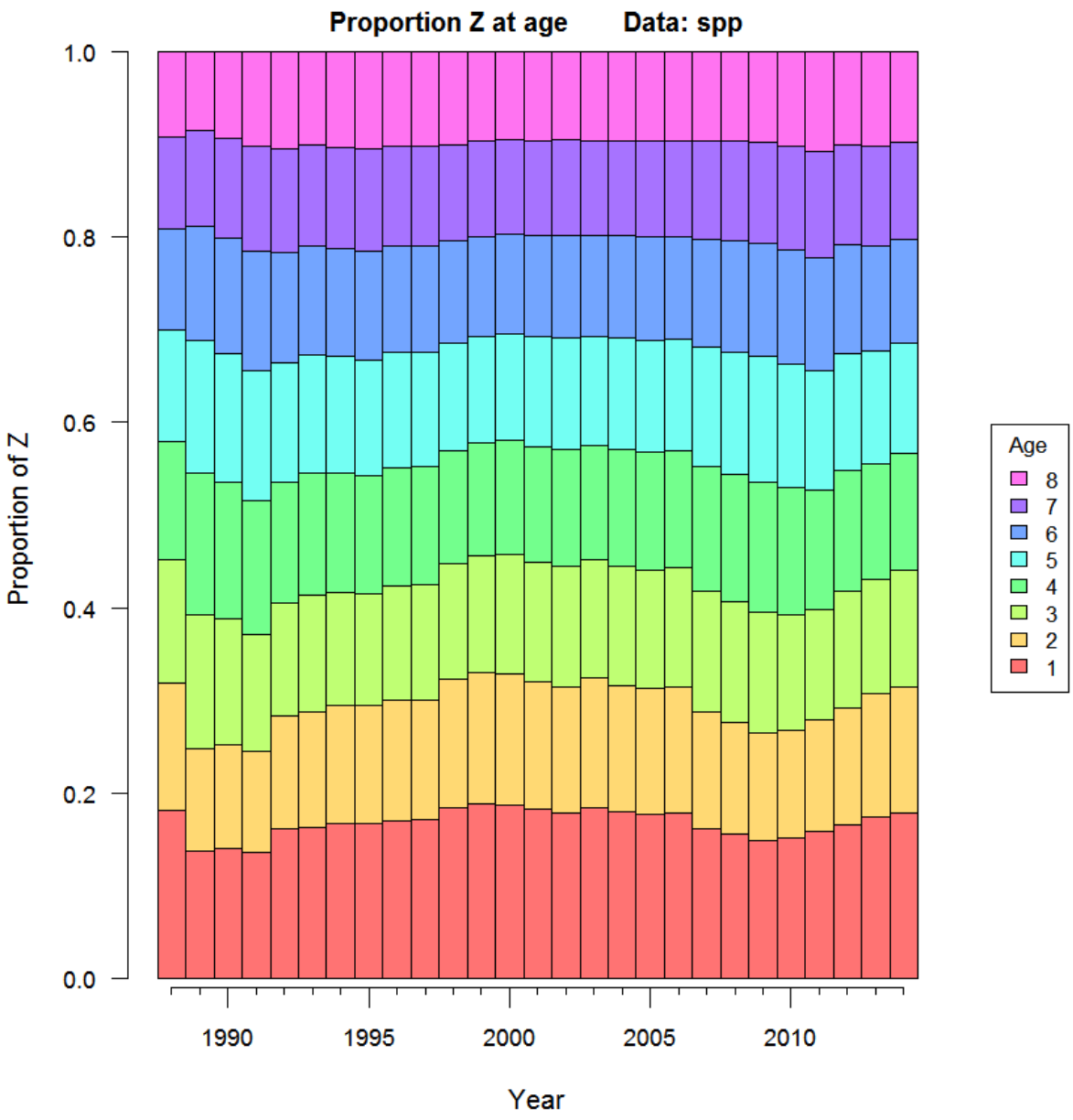

Fishery: acomp.HB Orange: underestimate Data: spp
0.46
0.34
0.23
0.11
0.05
Age class
8
7
6
5
4
3
2
1
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010

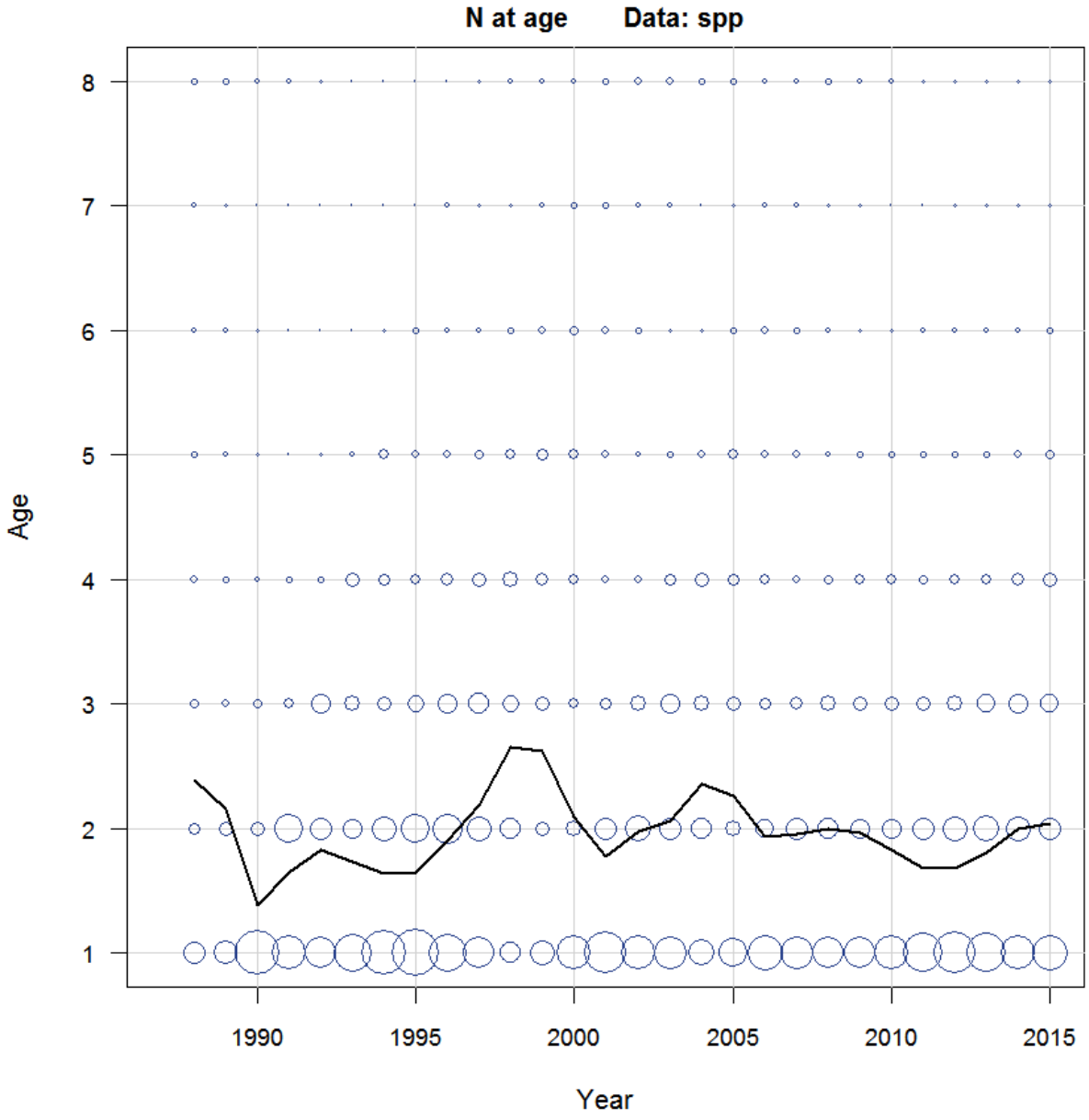

reprod Data: spp
reprod
0
2
4
6
1
2
3
4
5
6
7
8
Age

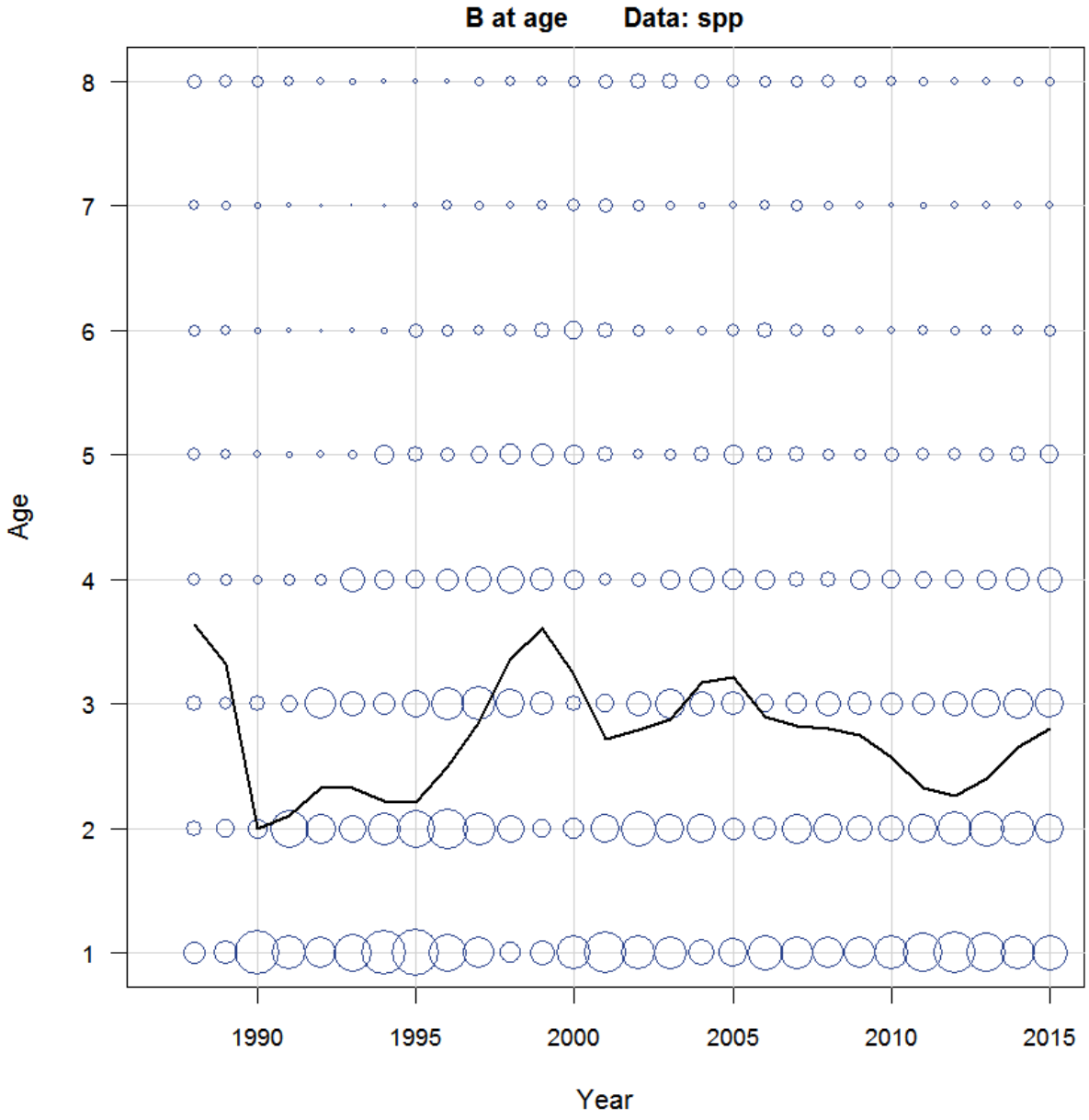

Nage.eq.init Data: spp
Nage.eq.init
0e+00
2e+06
4e+06
6e+06
8e+06
1
2
3
4
5
6
7
8
Age

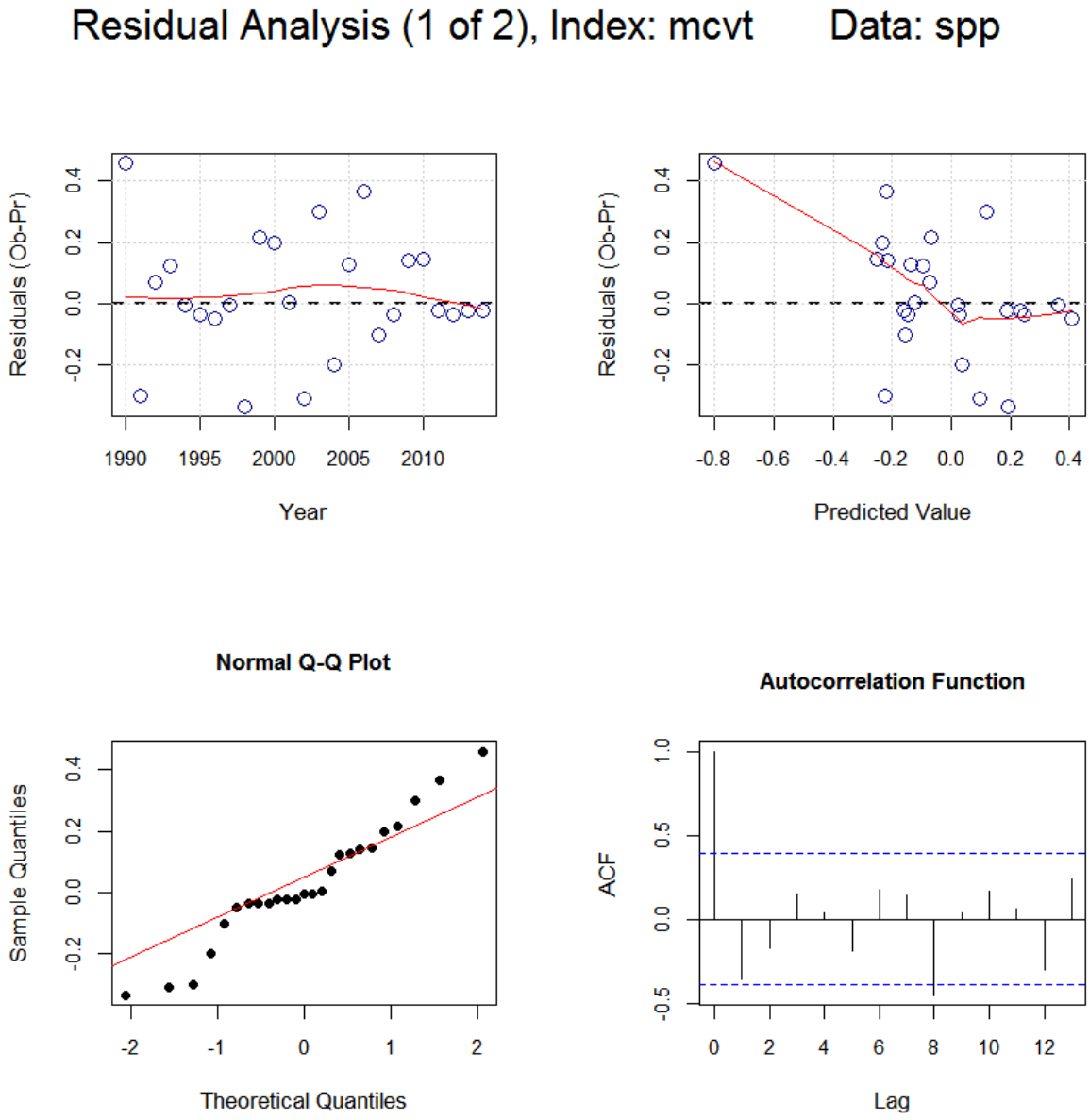

log.Nage.init.dev Data: spp
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
log.Nage.init.dev
1
2
3
4
5
6
7
8
Age


Proportion F at age Data: spp
Proportion of F
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year
Age
8
7
6
5
4
3
2
1

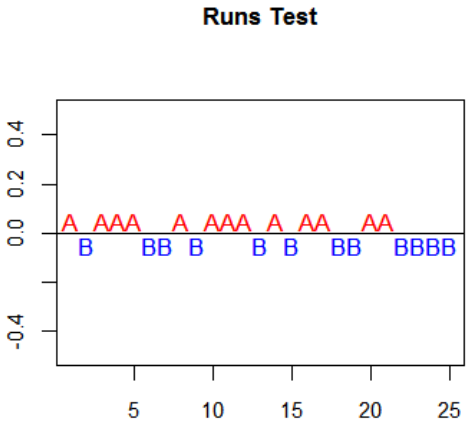

Proportion Z at age Data: spp
Proportion of Z
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year
Age
8
7
6
5
4
3
2
1

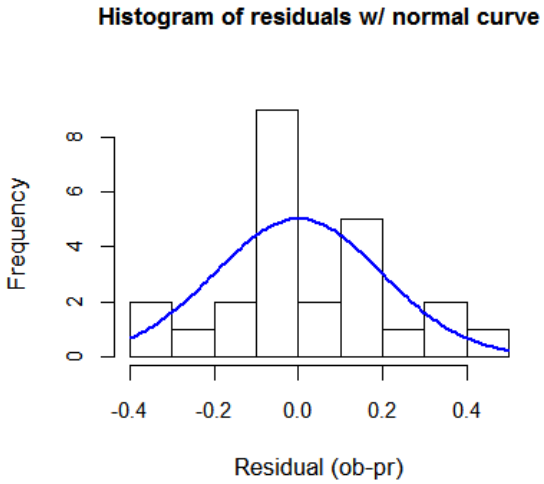

N at age Data: spp
Age
8
7
6
5
4
3
2
1
1990
1995
2000
2005
2010
2015
Year


B at age Data: spp
Age
Year
8
7
6
5
4
3
2
1
1990
1995
2000
2005
2010
2015

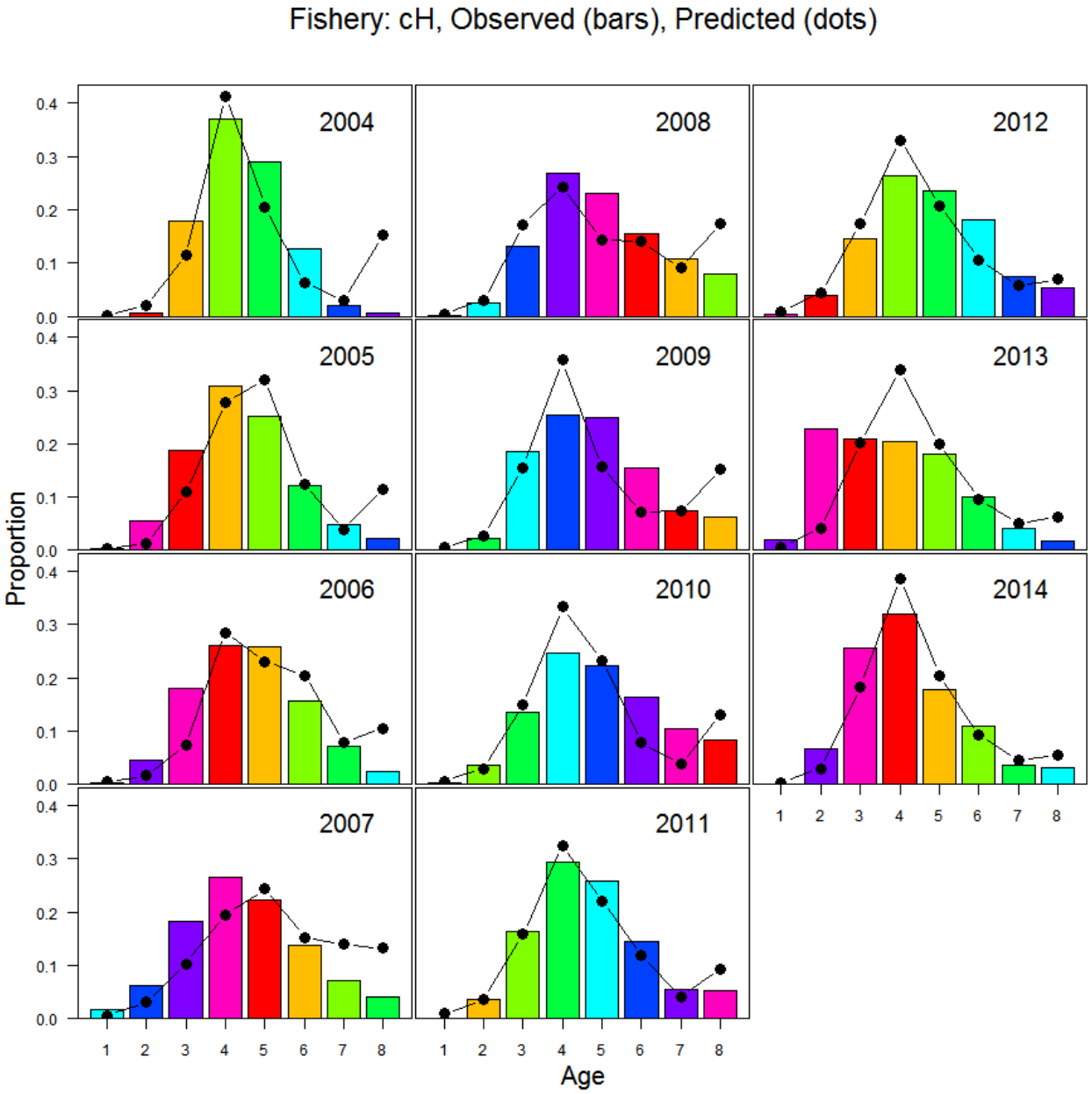

Residual Analysis (1 of 2), Index: mcvt Data: spp
Residuals (Ob-Pr)
Year
Predicted Value
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Autocorrelation Function
ACF
Lag

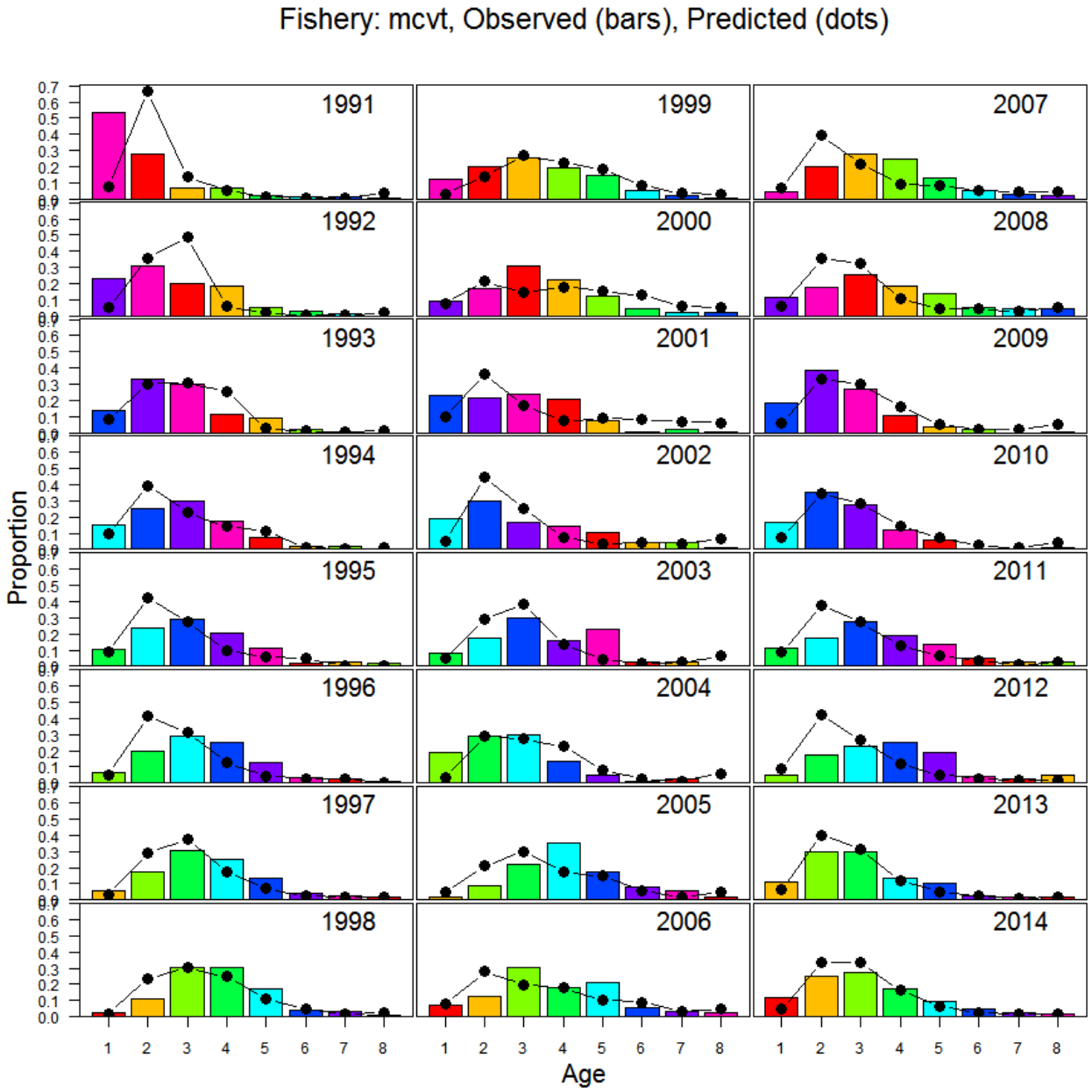
Residual Analysis (2 of 2), Index: mcvt Data: spp

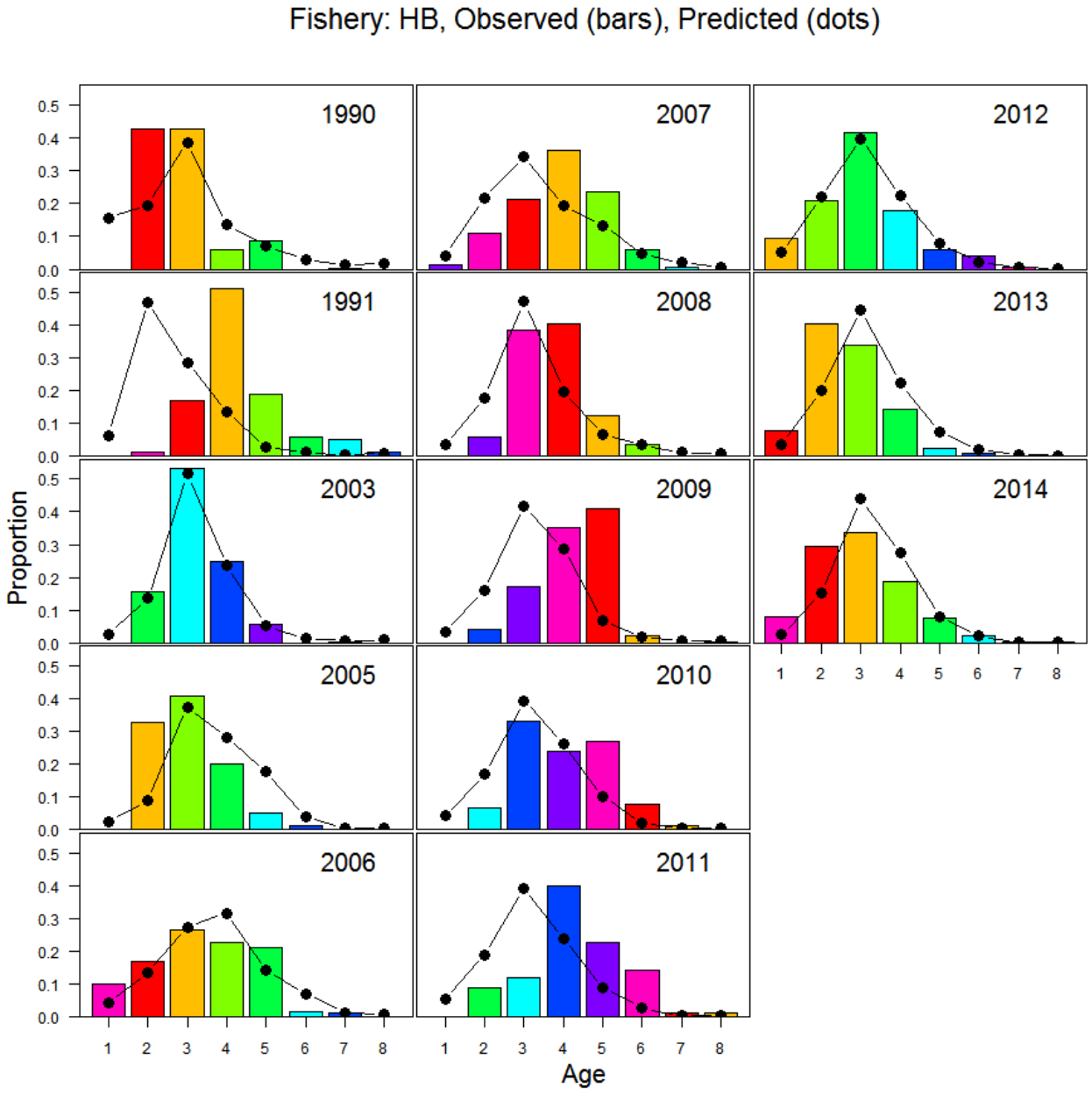


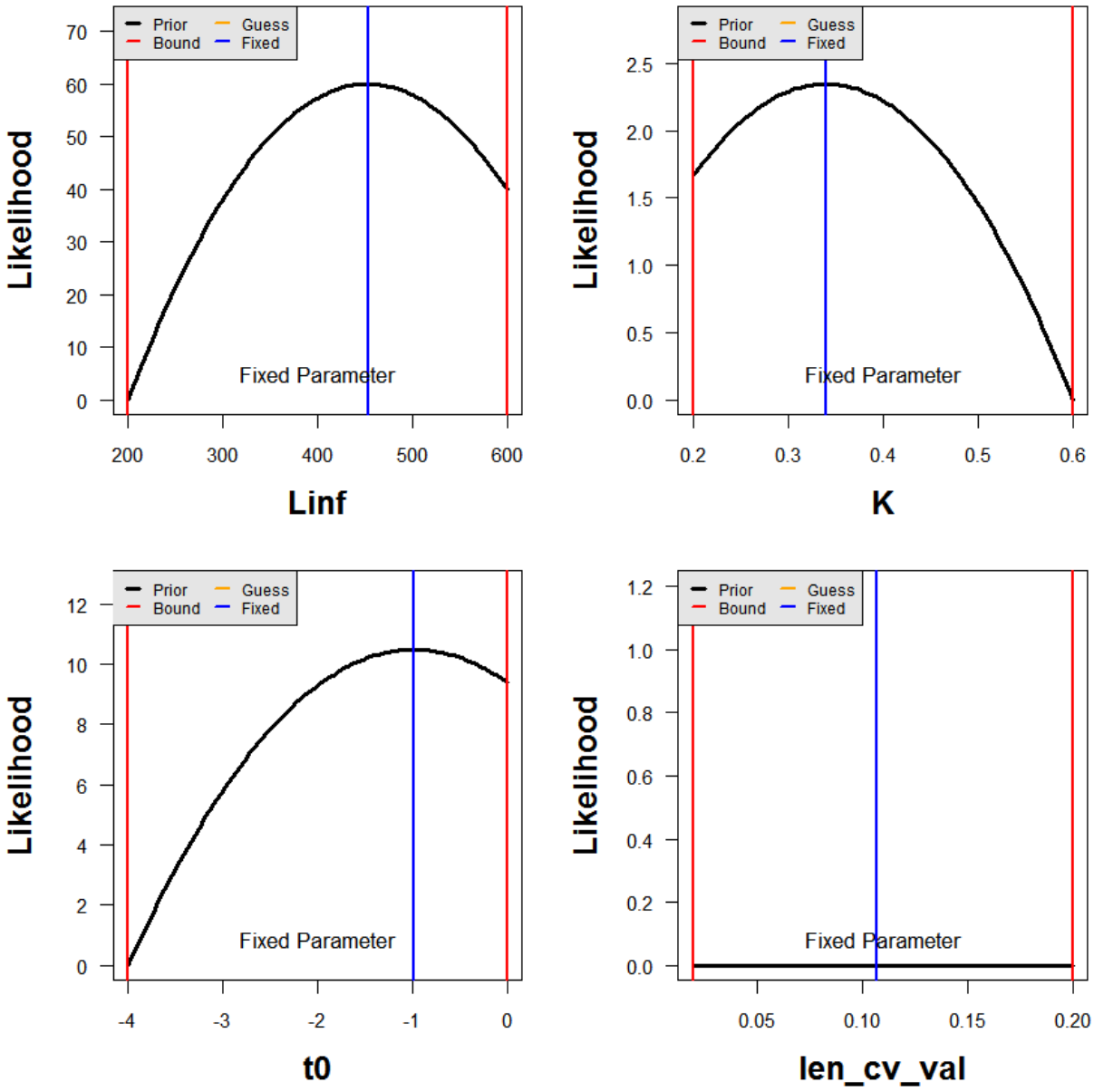


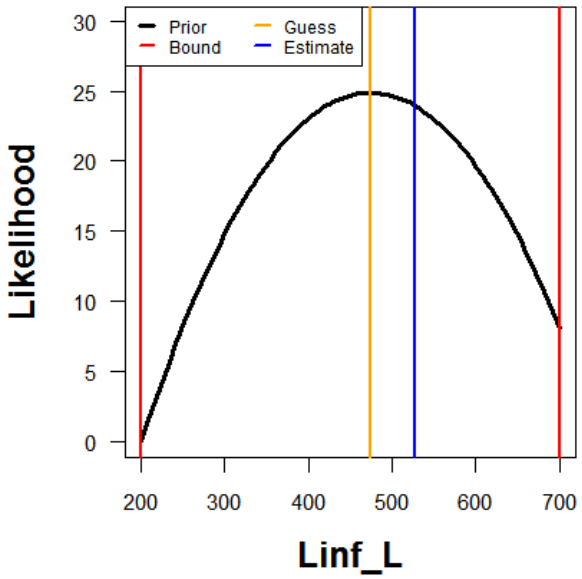
	P-values
Breusch-Pagan test for heteroskedasticity:	0.888
Harrison-McCabe test for heteroskedasticity:	0.553
sch-Godfrey test for higher-order serial correlation:	0
in-Watson test for autocorrelation of disturbances:	0
Lilliefors (Kolmogorov-Smirnov) test for normality:	0.175
Anderson-Darling test for normality:	0.2175
Pearson chi-square test for normality:	0.0483
Shapiro-Wilk test for normality:	0.4236
ps-Perron test for null hypothesis x has a unit root:	0.01
Runs test:	0.8315

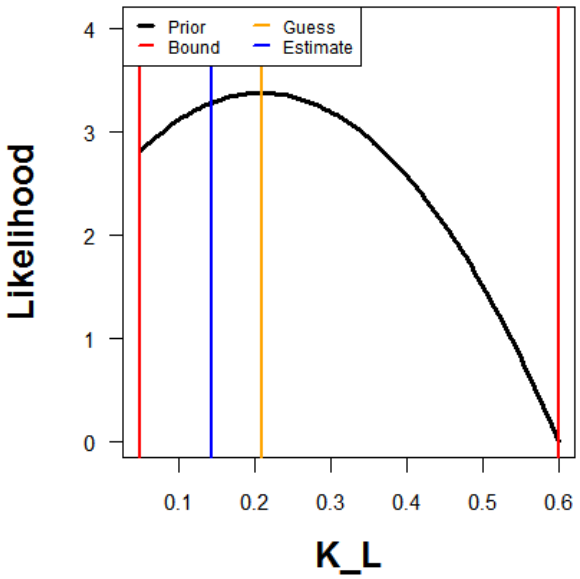

Fishery: cH, Observed (bars), Predicted (dots)
2004
2008
2012
2005
2009
2013
2006
2010
2014
2007
2011
Proportion
Age
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
8

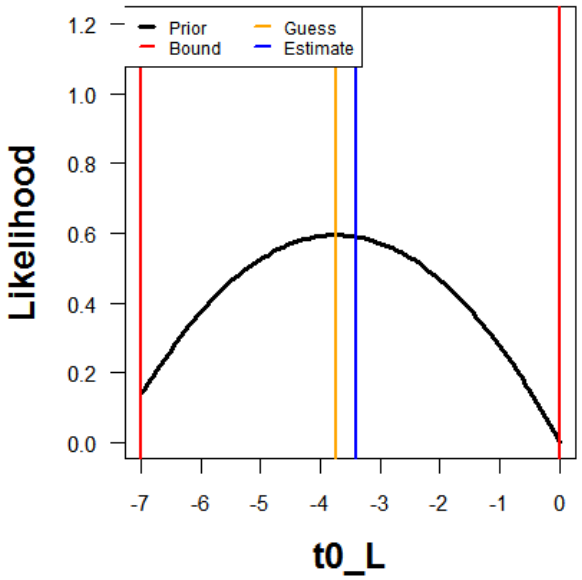

Fishery: mcvt, Observed (bars), Predicted (dots)
Proportion
Age
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

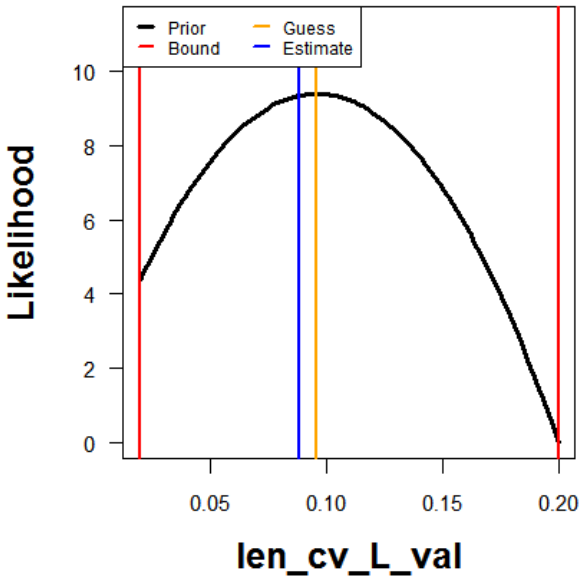

Fishery: HB, Observed (bars), Predicted (dots)
1990
2007
2012
1991
2008
2013
2003
2009
2014
2005
2010
2006
2011
Proportion
Age

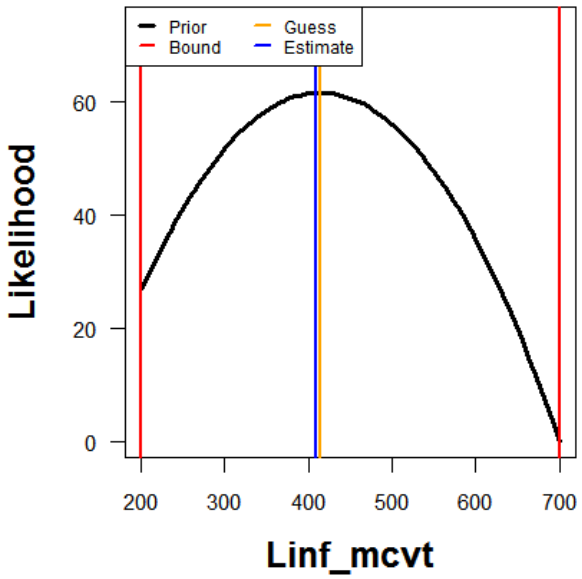

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
Linf
K
t0
len_cv_val

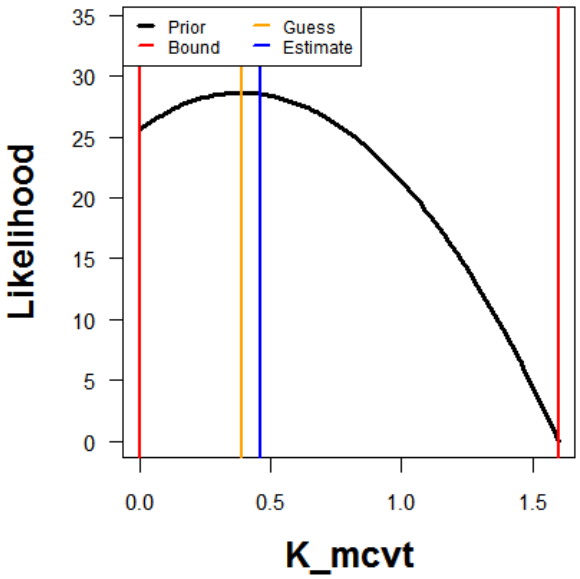

Prior
Bound
Guess
Estimate
Likelihood
Linf_L

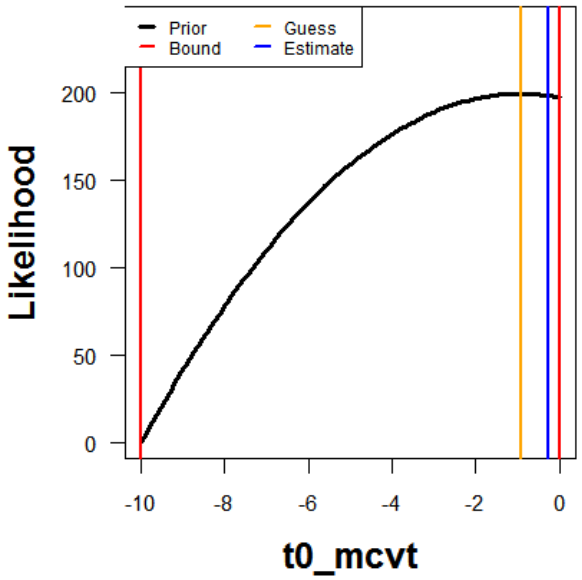

Prior
Bound
Guess
Estimate
Likelihood
K_L

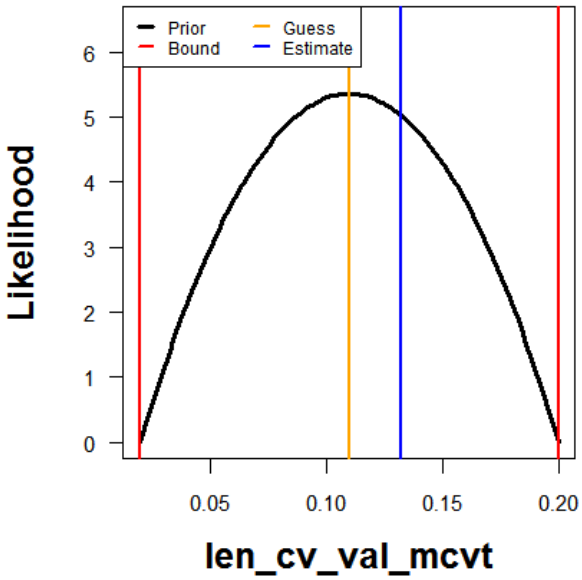

Prior
Bound
Guess
Estimate
Likelihood
t0_L

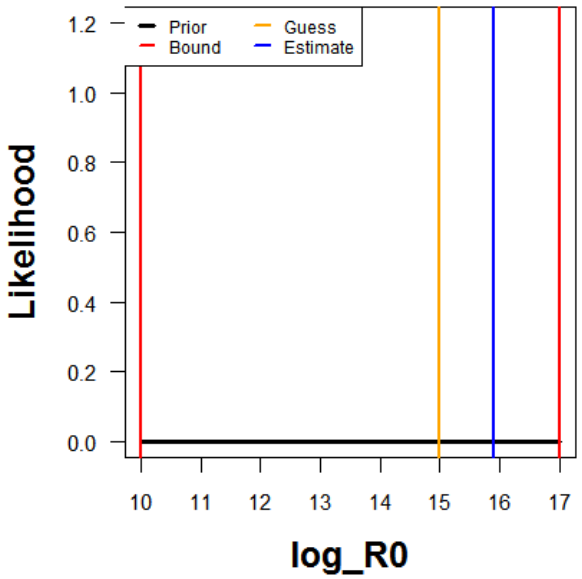

Prior
Bound
Guess
Estimate
Likelihood
len_cv_L_val

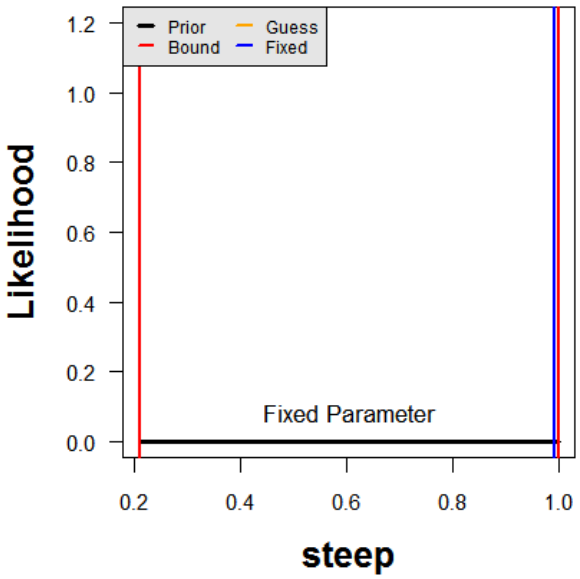

Prior
Bound
Guess
Estimate
Likelihood
0
20
40
60
200
300
400
500
600
700
Linf_mcvt

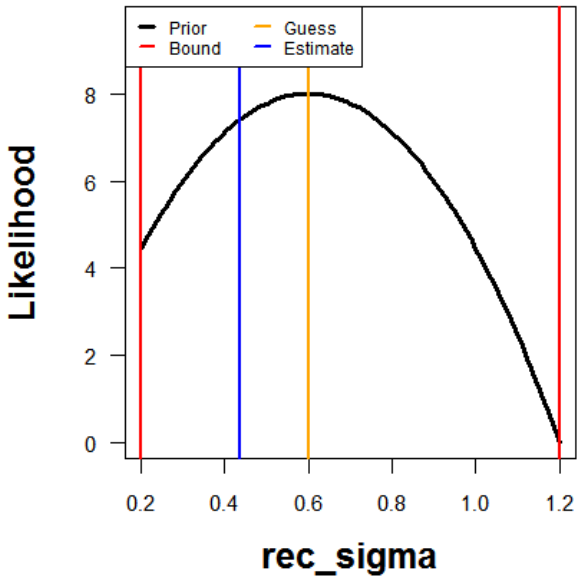

Prior
Bound
Guess
Estimate
Likelihood
0
5
10
15
20
25
30
35
0.0
0.5
1.0
1.5
K_mcvt

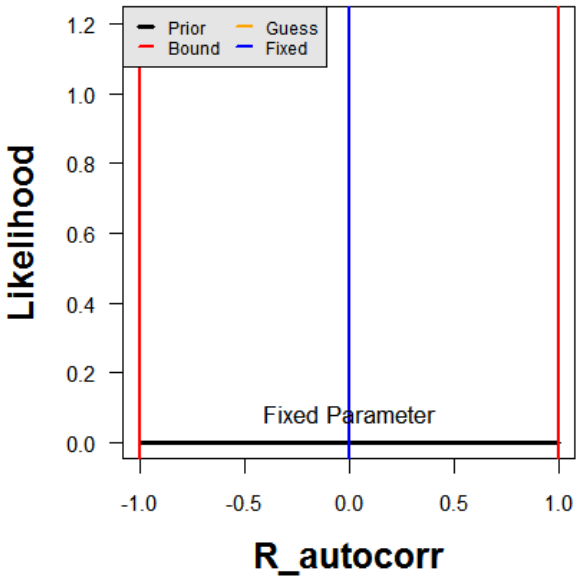

Prior
Bound
Guess
Estimate
Likelihood
0
50
100
150
200
-10
-8
-6
-4
-2
0
t0_mcvt

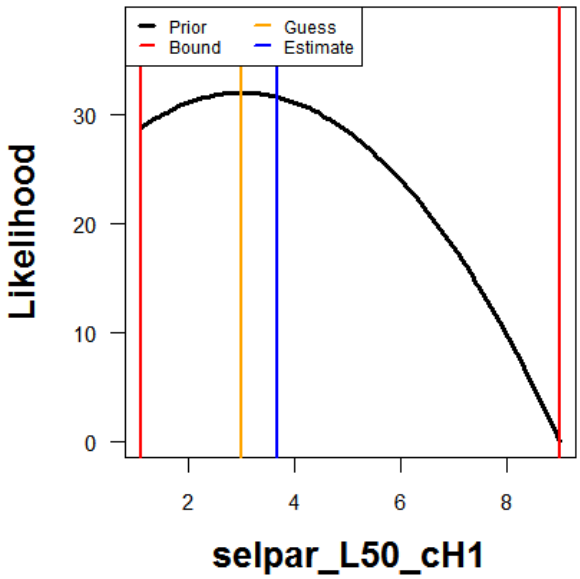

Prior
Bound
Guess
Estimate
Likelihood
0
1
2
3
4
5
6
0.05
0.10
0.15
0.20
len_cv_val_mcvt

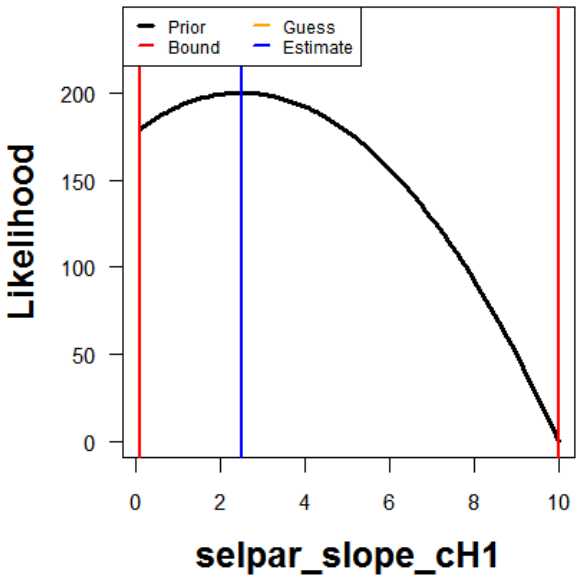

Prior
Bound
Guess
Estimate
Likelihood
log_R0

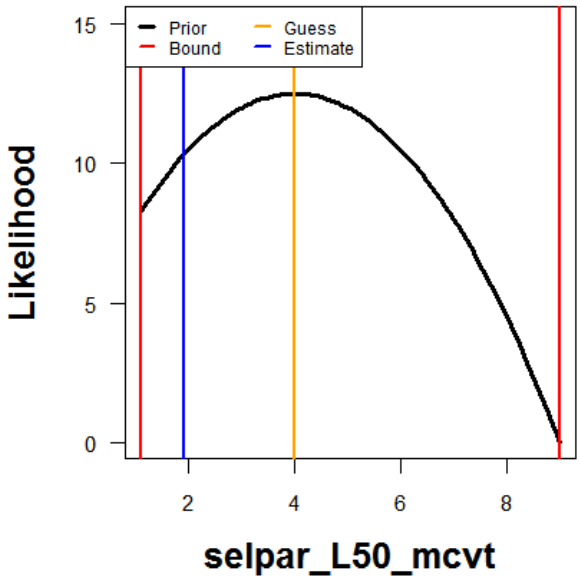

Prior
Bound
Guess
Fixed
Likelihood
Fixed Parameter
steep

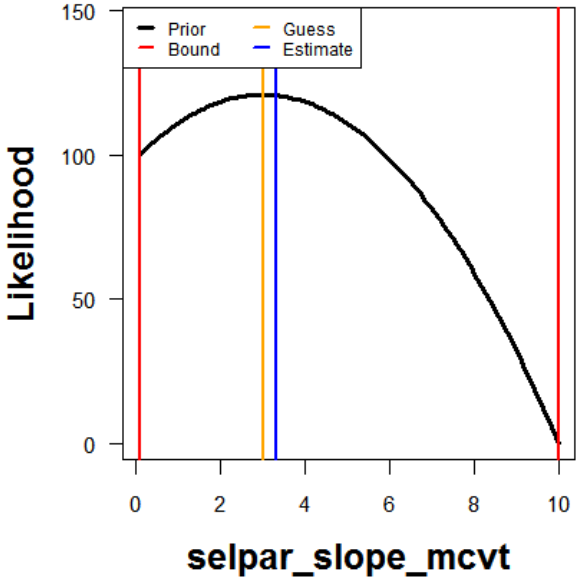

Prior
Bound
Guess
Estimate
Likelihood
rec_sigma

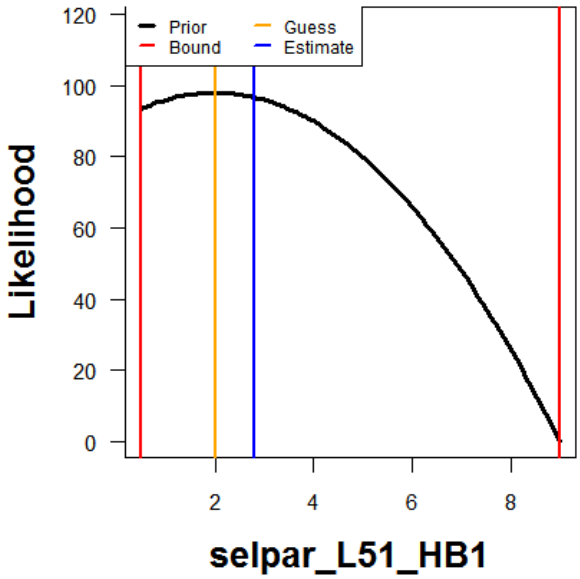

Prior
Bound
Guess
Fixed
Likelihood
Fixed Parameter
R_autocorr

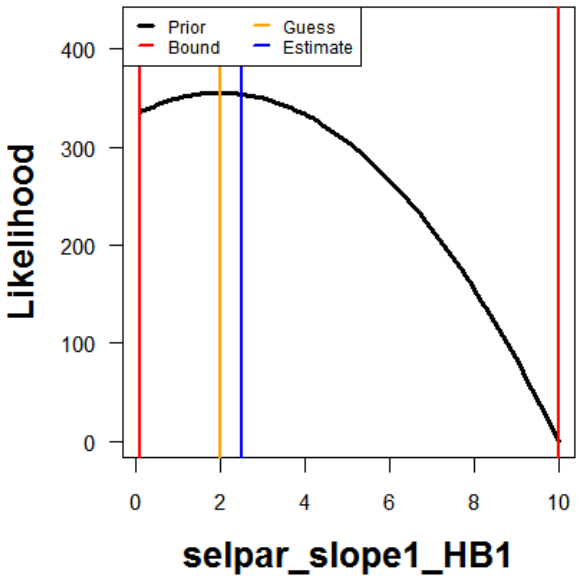

Prior
Bound
Guess
Estimate
Likelihood
selpar_L50_cH1

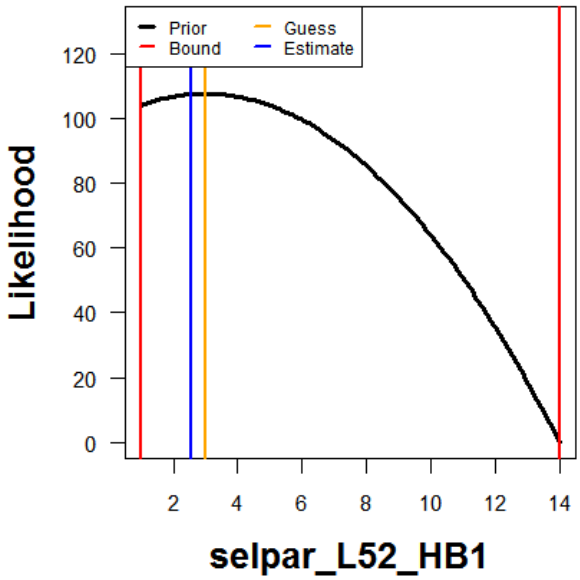

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_cH1

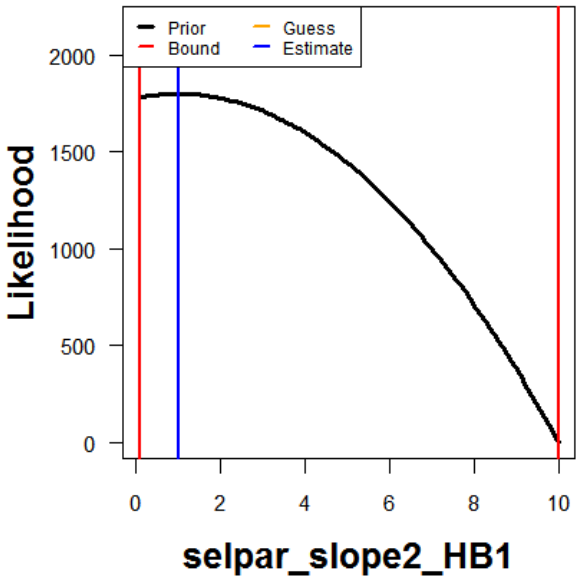

Prior
Bound
Guess
Estimate
Likelihood
selpar_L50_mcvt

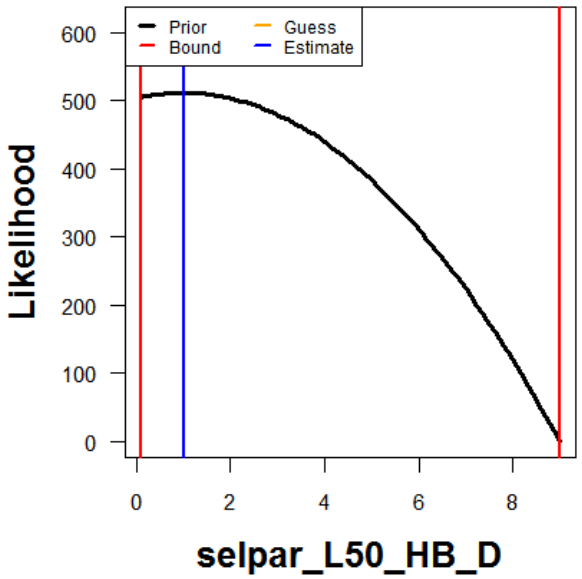

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_mcvt

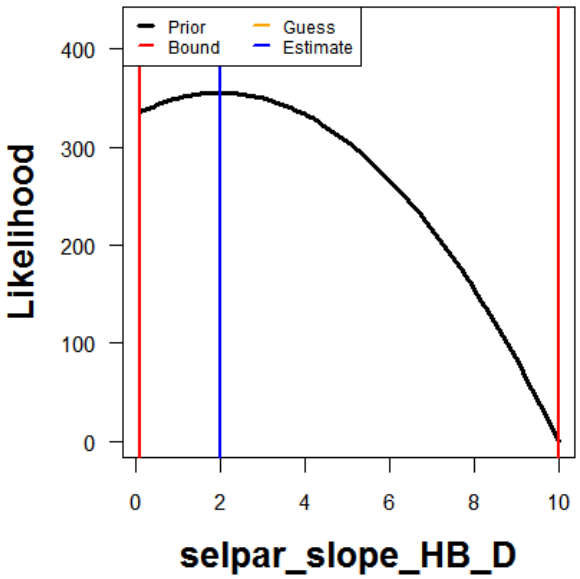

Prior
Bound
Guess
Estimate
Likelihood
selpar_L51_HB1

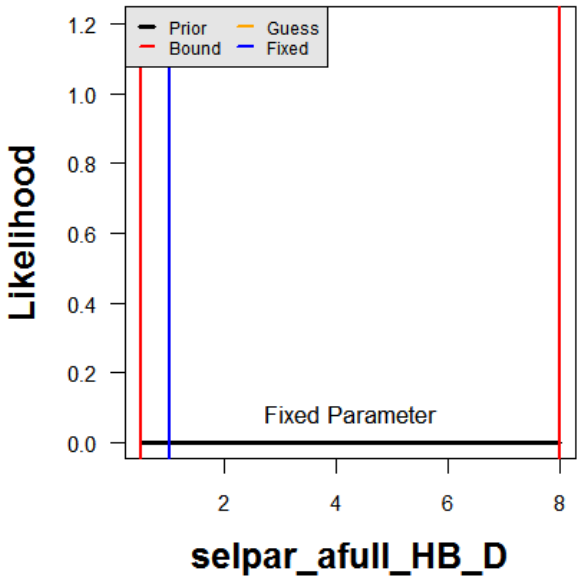

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_HB1

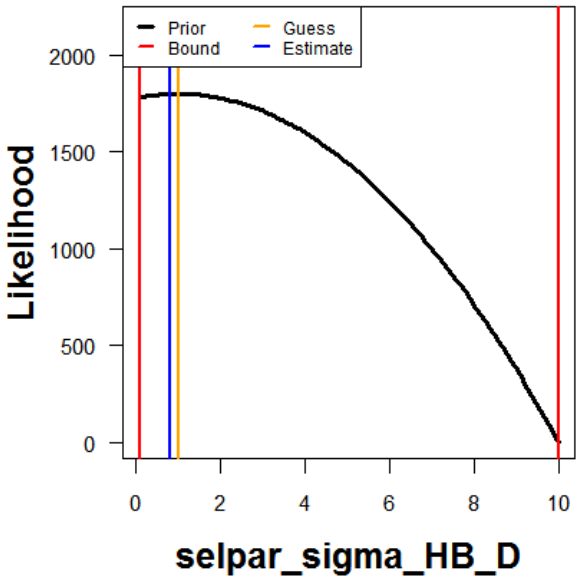

Prior
Bound
Guess
Estimate
Likelihood
selpar_L52_HB1

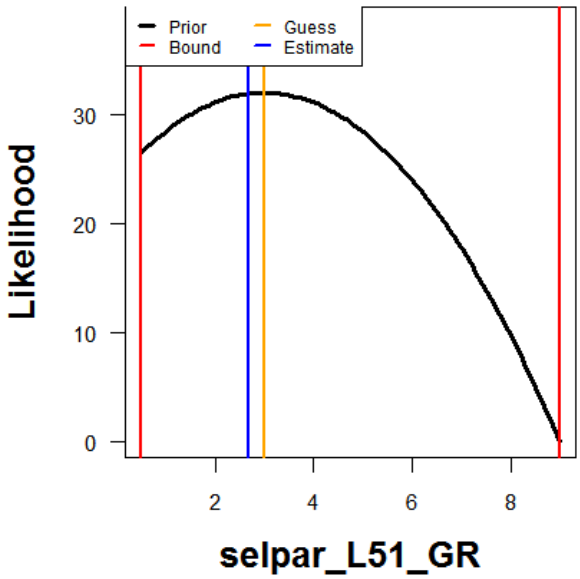

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB1

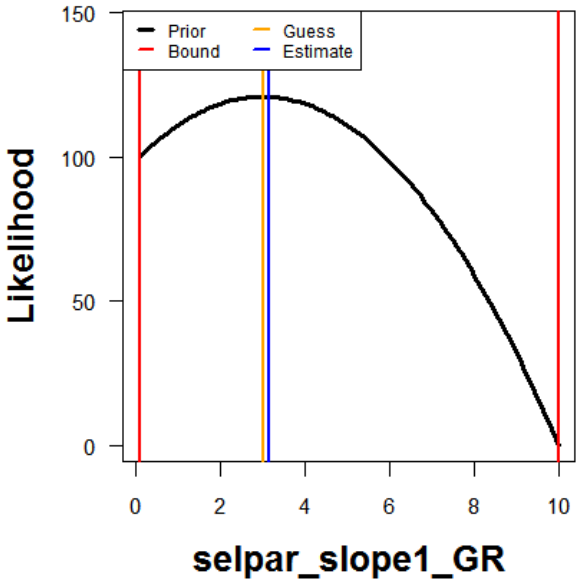

Prior
Bound
Guess
Estimate
Likelihood
selpar_L50_HB_D

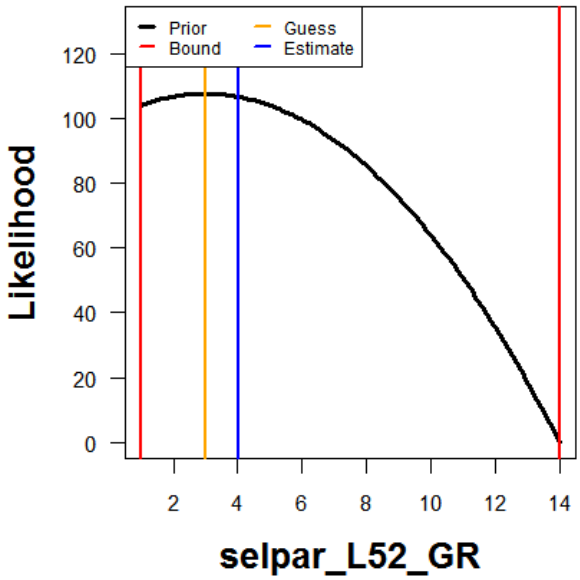

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_HB_D

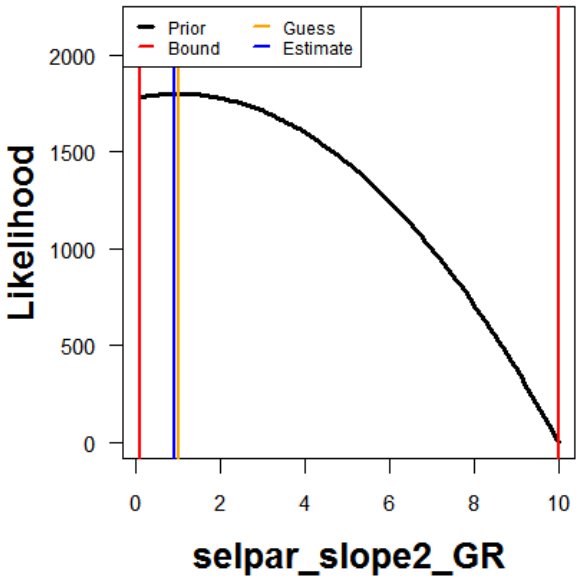

Prior
Bound
Guess
Fixed
Fixed Parameter
Likelihood
selpar_afull_HB_D

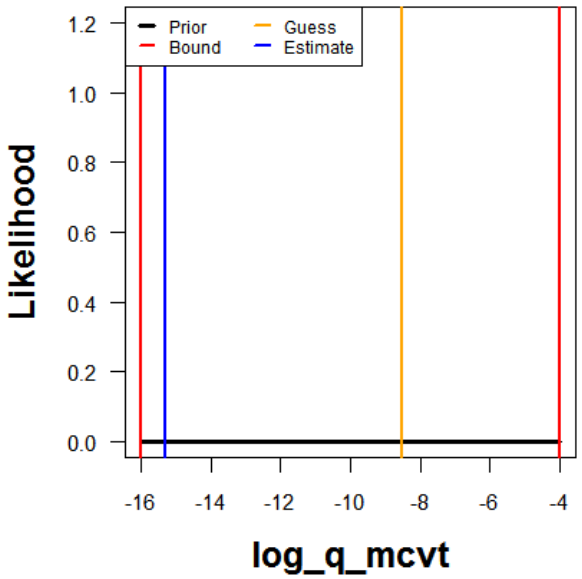

Prior
Bound
Guess
Estimate
Likelihood
selpar_sigma_HB_D

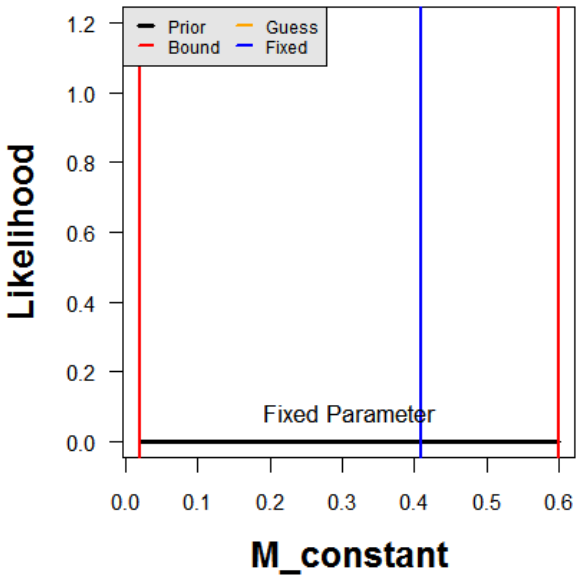

Prior
Bound
Guess
Estimate
Likelihood
0
10
20
30
2
4
6
8
selpar_L51_GR

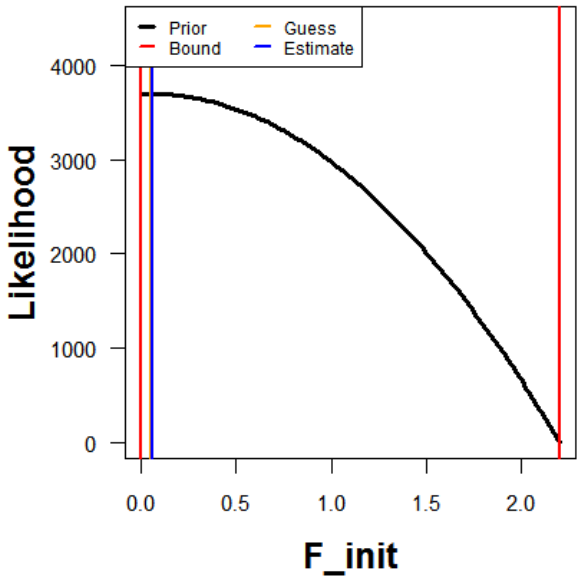

Prior
Bound
Guess
Estimate
Likelihood
0
50
100
150
0
2
4
6
8
10
selpar_slope1_GR

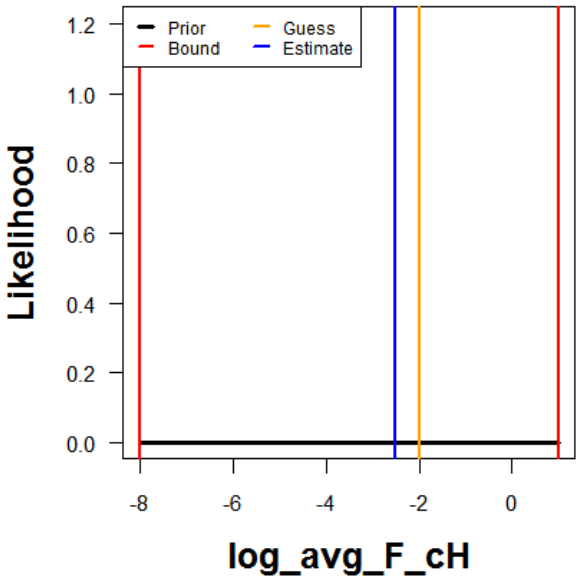

Prior
Bound
Guess
Estimate
Likelihood
0
20
40
60
80
100
120
2
4
6
8
10
12
14
selpar_L52_GR

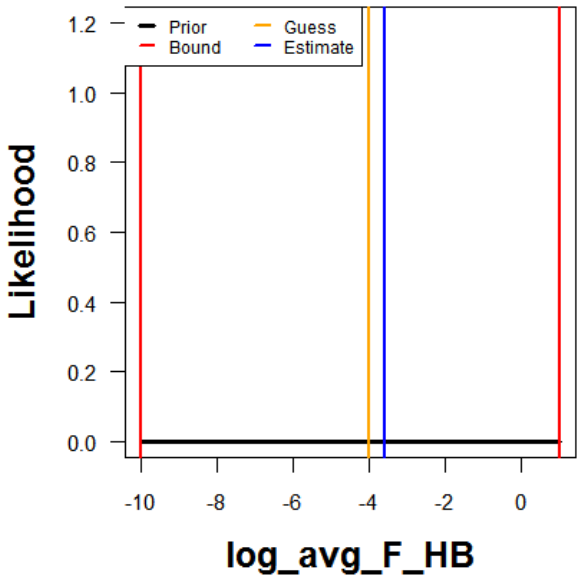

Prior
Bound
Guess
Estimate
Likelihood
0
500
1000
1500
2000
0
2
4
6
8
10
selpar_slope2_GR

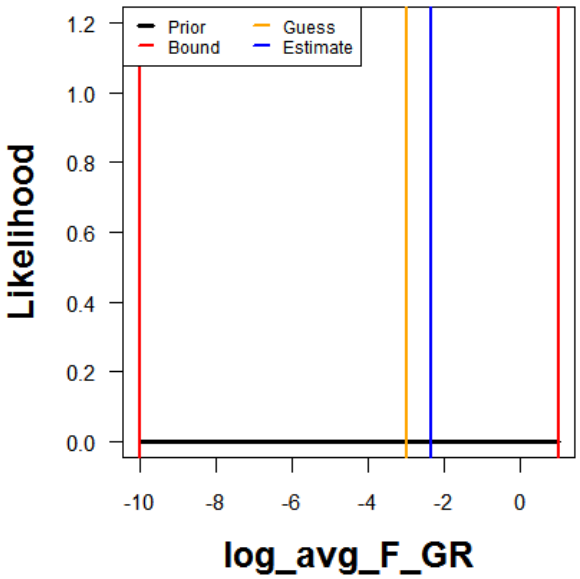

Prior
Bound
Guess
Estimate
Likelihood
log_q_mcvt

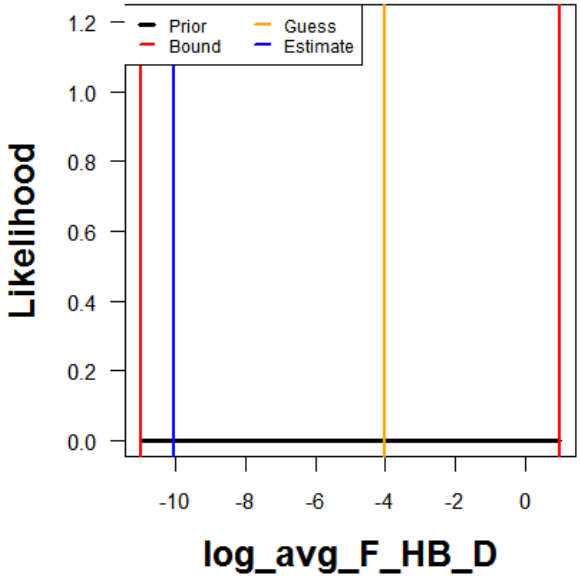

Prior
Bound
Guess
Fixed
Fixed Parameter
Likelihood
M_constant

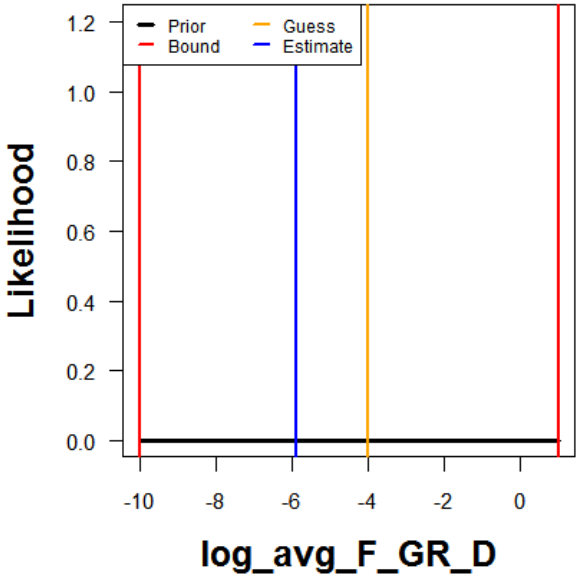

Prior
Bound
Guess
Estimate
Likelihood
F_init


Prior
Bound
Guess
Estimate
Likelihood
log_avg_F_cH

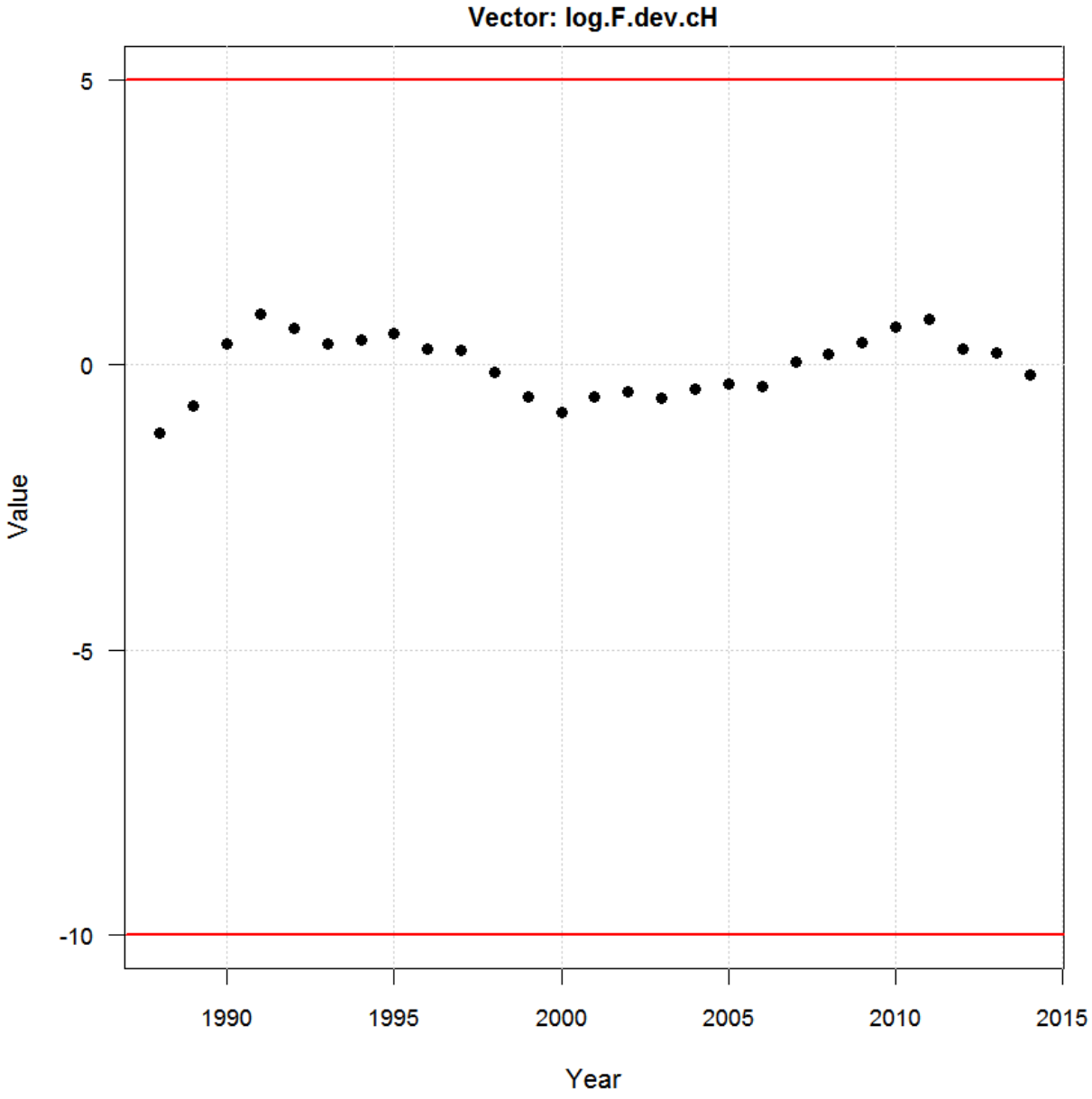

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_HB

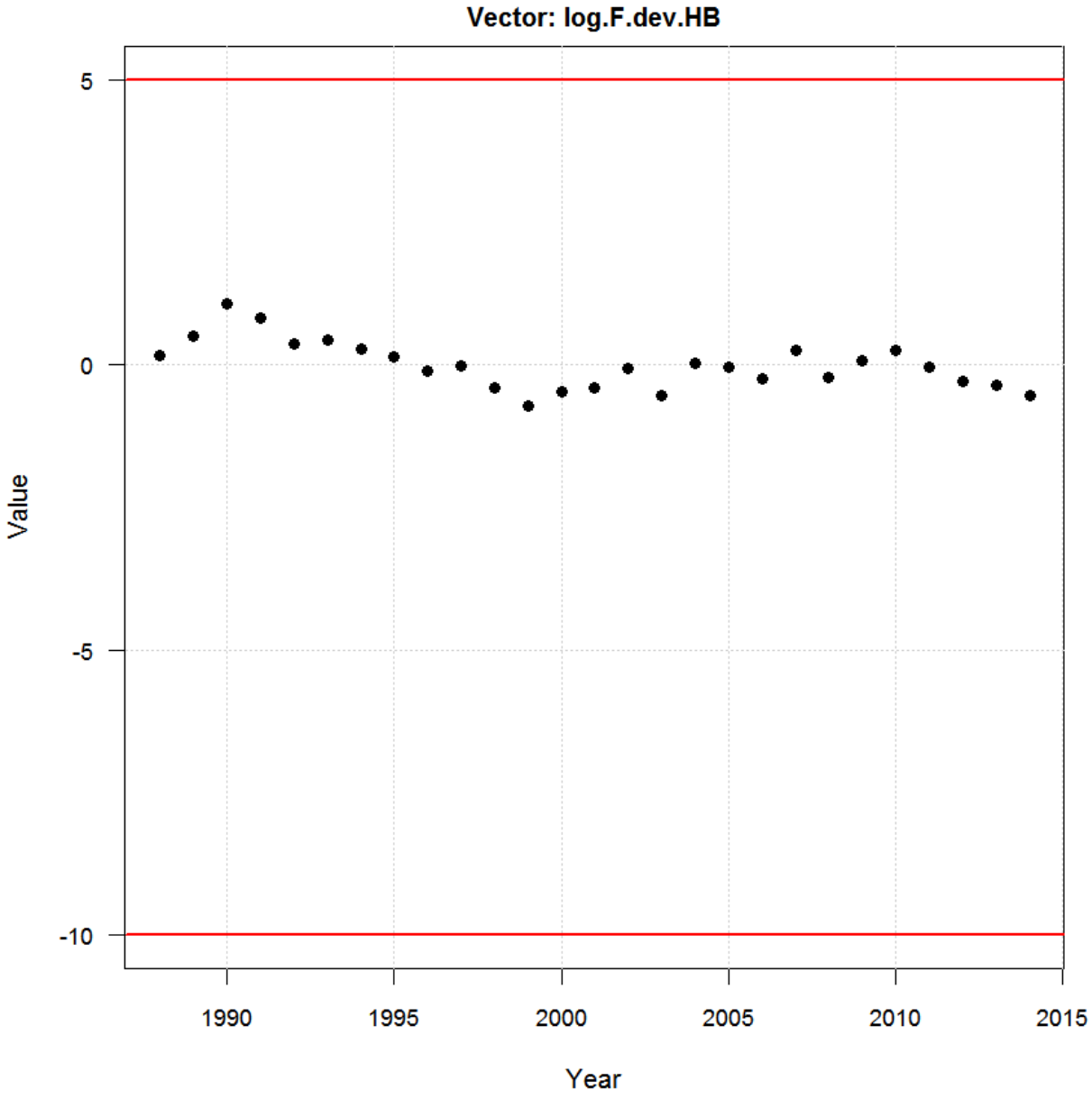

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_GR

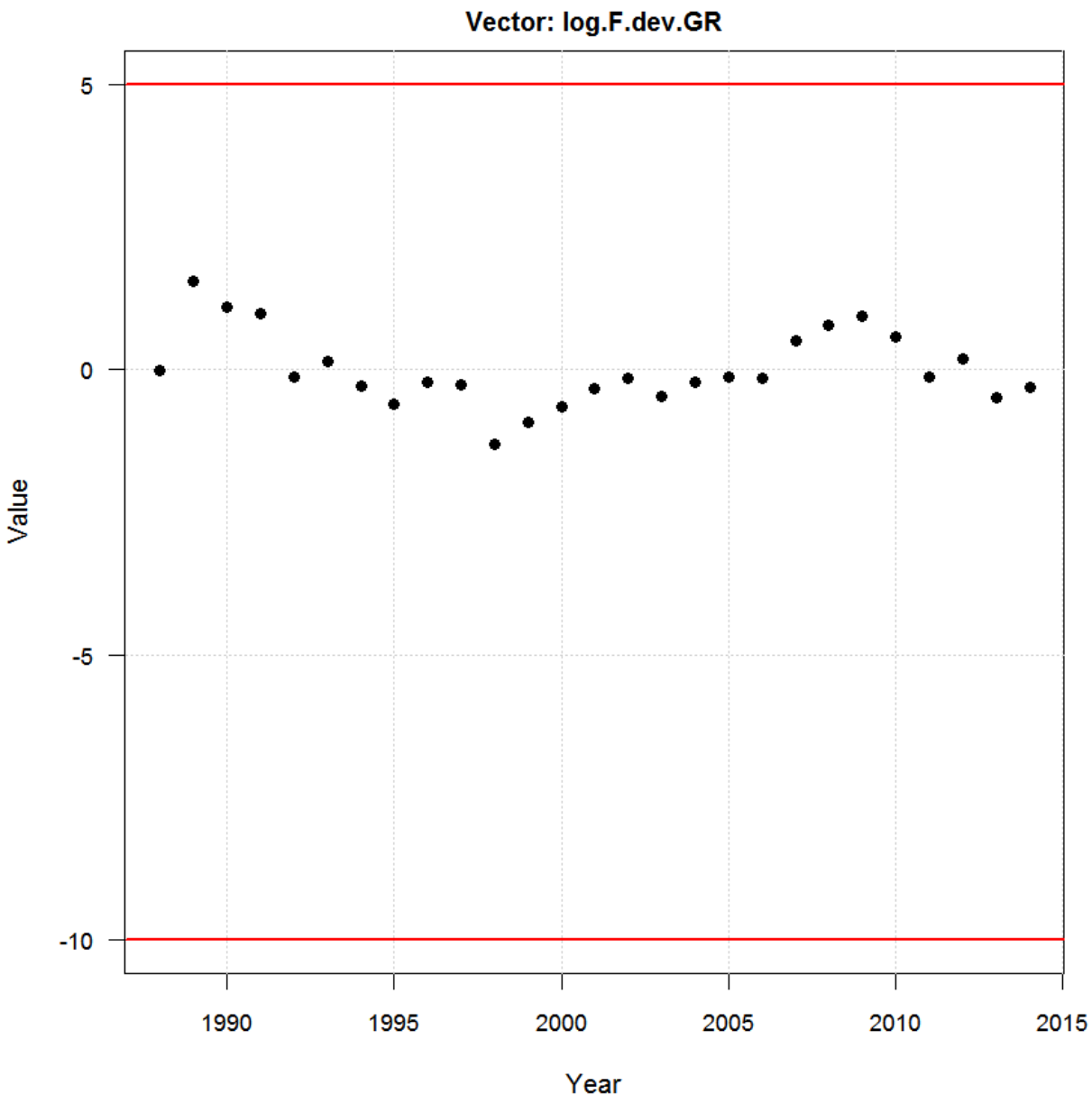

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_HB_D

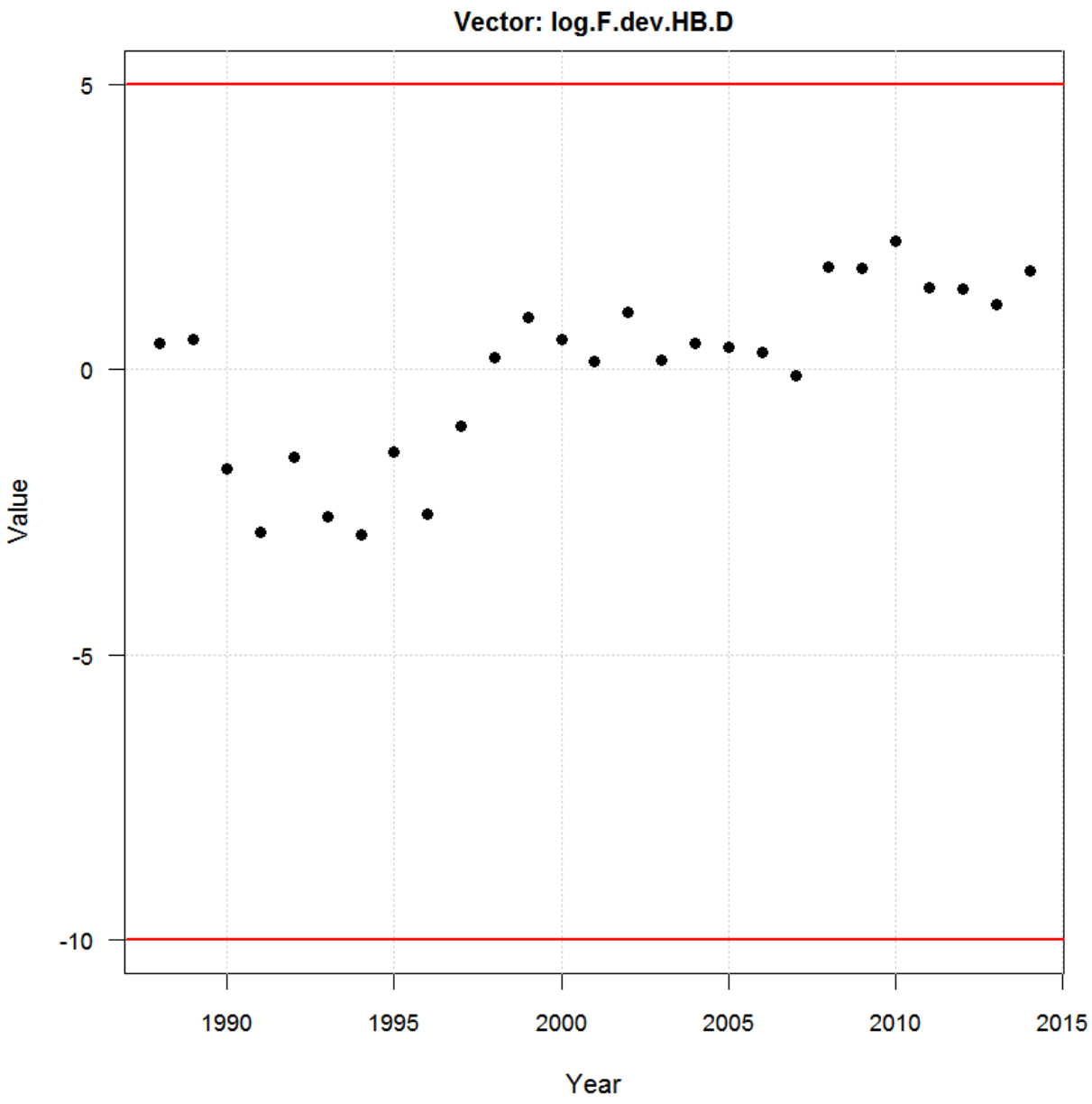

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_GR_D

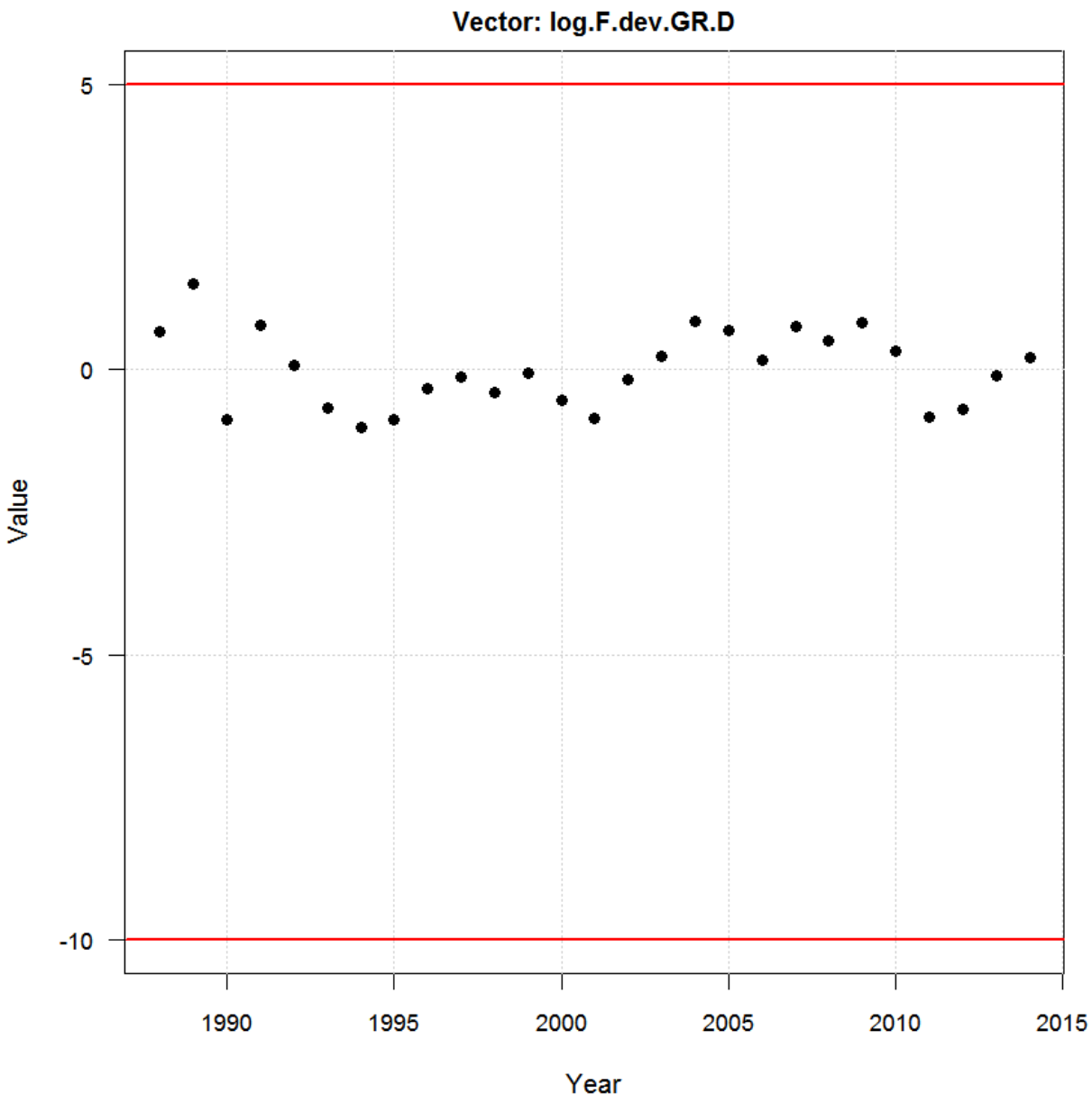

Vector: log.rec.dev
Value
4
2
0
-2
-4
1990
1995
2000
2005
2010
2015
Year

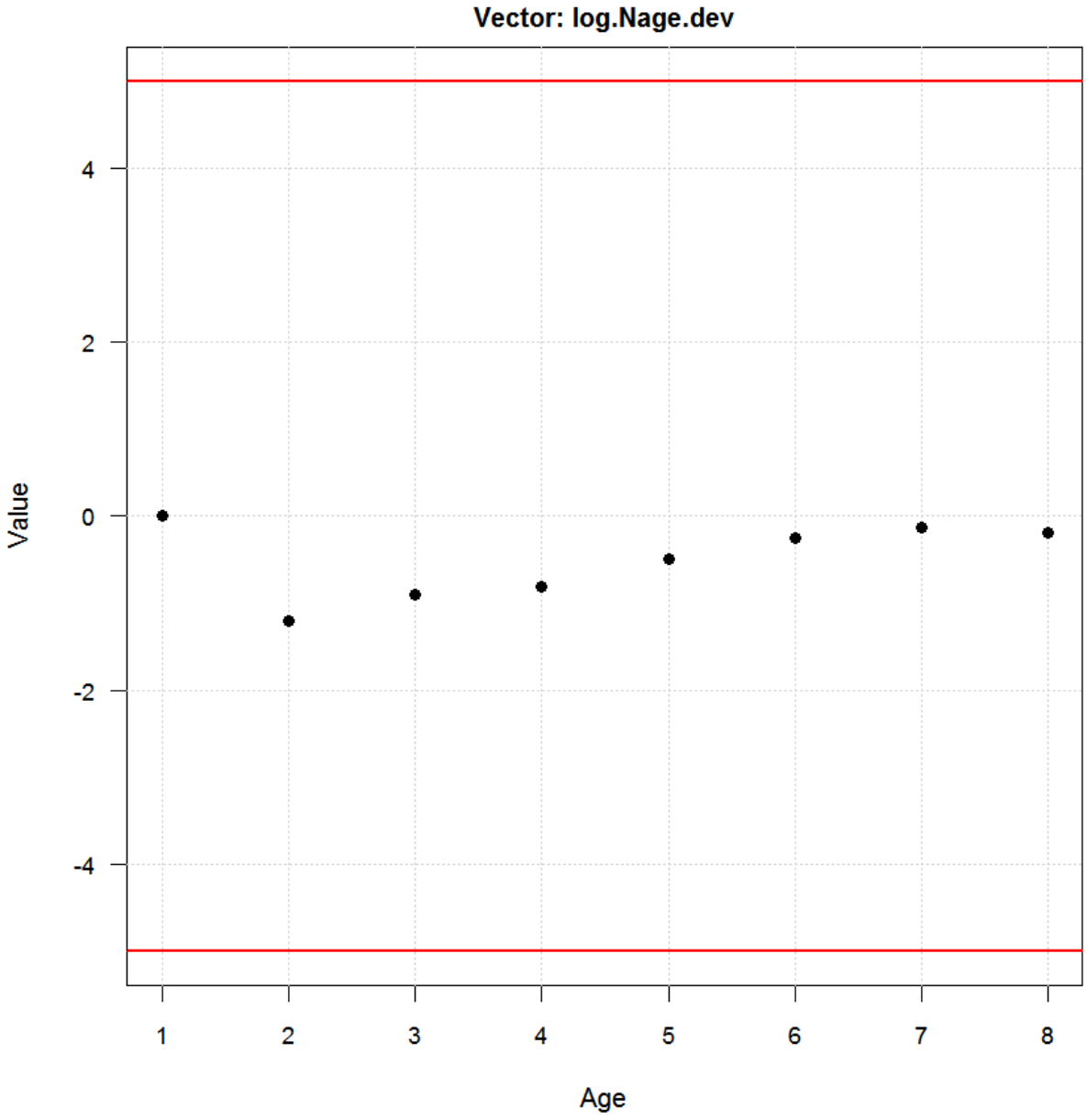

Vector: log.F.dev.cH
5
0
-5
-10
Value
1990
1995
2000
2005
2010
2015
Year


Vector: log.F.dev.HB
5
0
-5
-10
Value
1990
1995
2000
2005
2010
2015
Year


Vector: log.F.dev.GR
5
0
-5
-10
Value
1990
1995
2000
2005
2010
2015
Year


Vector: log.F.dev.HB.D
5
0
-5
-10
Value
1990
1995
2000
2005
2010
2015
Year


Vector: log.F.dev.GR.D
5
0
-5
-10
Value
1990
1995
2000
2005
2010
2015
Year


Vector: log.Nage.dev
Value
4
2
0
-2
-4
1
2
3
4
5
6
7
8
Age

ADMB tpl file for the BAM:

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

//##

//##  SEDAR 41 benchmark  Gray Triggerfish assessment October 2015

//##  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; //gt has no selectivity blocks

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

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

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

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

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

// Max F used in spr and msy calcs
init_number max_F_spr_msy;
// Total number of iterations for spr calcs
init_int n_iter_spr;
// Total number of iterations for msy calcs
int n_iter_msy;
// Number years at end of time series over which to average sector F's, for weighted selectivities
 LOCAL_CALCS
                        n_iter_msy=n_iter_spr;
 END_CALCS

init_int selpar_n_yrs_wgted;
// Bias correction (set to 1.0 for no bias correction or a negative value to compute from rec variance)
init_number set_BiasCor;

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

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

```
// Comm HL (Landings+Others+Discards; 1000 lb whole weight)
init_int styr_cH_L;
init_int endyr_cH_L;
init_vector obs_cH_L(styr_cH_L,endyr_cH_L);
init_vector cH_L_cv(styr_cH_L,endyr_cH_L);

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

// Comm 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);

//################MARMAP Chevron trap ##############################################
// SERFS chevron trap CPUE
init_int styr_mcvt_cpue;
init_int endyr_mcvt_cpue;
init_vector obs_mcvt_cpue(styr_mcvt_cpue,endyr_mcvt_cpue); //Observed CPUE
init_vector mcvt_cpue_cv(styr_mcvt_cpue,endyr_mcvt_cpue);  //CV of cpue

// SERFS chevron trap length compositions (3 cm bins)
```

```
init_int nyr_mcvt_lenc;
init_ivector yrs_mcvt_lenc(1,nyr_mcvt_lenc);
init_vector nsamp_mcvt_lenc(1,nyr_mcvt_lenc);
init_vector nfish_mcvt_lenc(1,nyr_mcvt_lenc);
init_matrix obs_mcvt_lenc(1,nyr_mcvt_lenc,1,nlenbins);

// SERFS chevron trap age compositions
init_int nyr_mcvt_agec;
init_ivector yrs_mcvt_agec(1,nyr_mcvt_agec);
init_vector nsamp_mcvt_agec(1,nyr_mcvt_agec);
init_vector nfish_mcvt_agec(1,nyr_mcvt_agec);
init_matrix obs_mcvt_agec(1,nyr_mcvt_agec,1,nages_agec);

//#################SEFIS video index ##############################################
// SEFIS video CPUE
//init_int styr_vid_cpue;
//init_int endyr_vid_cpue;
//init_vector obs_vid_cpue(styr_vid_cpue,endyr_vid_cpue); //Observed CPUE
//init_vector vid_cpue_cv(styr_vid_cpue,endyr_vid_cpue);  //CV of cpue

//####################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);//Observed CPUE
init_vector HB_cpue_cv(styr_HB_cpue,endyr_HB_cpue); //CV of 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);   //vector of observed landings by year
init_vector HB_L_cv(styr_HB_L,endyr_HB_L);    //vector of CV of landings by year

// HB landings 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 landings 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);

// 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 discard length compositions
init_int nyr_HB_D_lenc;
init_ivector yrs_HB_D_lenc(1,nyr_HB_D_lenc);
init_vector nsamp_HB_D_lenc(1,nyr_HB_D_lenc);
init_vector nfish_HB_D_lenc(1,nyr_HB_D_lenc);
```

```
init_matrix obs_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);

// !! cout << styr_HB_D << endl;
//####################General Recreational fleet ###################################
// GR (MRIP) CPUE
init_int styr_GR_cpue;
init_int endyr_GR_cpue;
init_vector obs_GR_cpue(styr_GR_cpue,endyr_GR_cpue);//Observed CPUE
init_vector GR_cpue_cv(styr_GR_cpue,endyr_GR_cpue); //CV of cpue

// GR (MRIP) landings (1000s fish)
init_int styr_GR_L;
init_int endyr_GR_L;
init_vector obs_GR_L(styr_GR_L,endyr_GR_L);   //vector of observed landings by year
init_vector GR_L_cv(styr_GR_L,endyr_GR_L);    //vector of CV of landings by year

// GR (MRIP) landings 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 landings age compositions (single pooled age comp over 2004-2005 MRIP CB mode; weighted by number trips)
// 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);


// 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);


//  !! cout << obs_GR_released << endl;
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--
><>
//-- BAM DATA_SECTION: parameter section
//--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--
><>


//###################Single Parameter values and initial guesses ########################################
// Von Bert parameters in mm FL for the popn
init_vector set_Linf(1,7);
init_vector set_K(1,7);
init_vector set_t0(1,7);
//CV of length at age or its standard error for the popn
init_vector set_len_cv(1,7);


// FISHERY LANDINGS   Von Bert parameters in mm FL for all FD fish
init_vector set_Linf_L(1,7);
init_vector set_K_L(1,7);
init_vector set_t0_L(1,7);
//CV of length at age landings
init_vector set_len_cv_L(1,7);
```

```
// FISHERY INDEPENDENT  Von Bert parameters in mm FL mm all FI (SERFS chevron trap) fish
init_vector set_Linf_mcvt(1,7);
init_vector set_K_mcvt(1,7);
init_vector set_t0_mcvt(1,7);
//CV of length at age fishery-independent trap
init_vector set_len_cv_mcvt(1,7)

// Scalar used only for computing MSST. This might eventually be based on 0.25: MSST=(1-0.25)SSBmsy
init_vector set_M_constant(1,7);

// Spawner-recruit parameters (Initial guesses or fixed values)
init_vector set_steep(1,7);         //recruitment steepness
init_vector set_log_R0(1,7);        //recruitment R0
init_vector set_R_autocorr(1,7);    //recruitment autocorrelation
init_vector set_rec_sigma(1,7);     //recruitment standard deviation in log space

// Initial guesses or fixed values of estimated selectivity parameters
// Commercial handline selectivity
init_vector set_selpar_L50_cH1(1,7);
init_vector set_selpar_slope_cH1(1,7);

// init_vector set_selpar_L50_cH2(1,7);   //if using selectivity blocks
// init_vector set_selpar_slope_cH2(1,7); //if using selectivity blocks

// init_vector set_selpar_L50_cH3(1,7);   //if using selectivity blocks
// init_vector set_selpar_slope_cH3(1,7); //if using selectivity blocks

// SERFS chevron trap selectivity
init_vector set_selpar_L50_mcvt(1,7);
```

```
init_vector set_selpar_slope_mcvt(1,7);

// Headboat selectivity (set up as 2 blocks to allow for separate selectivities for specific sets of years)
init_vector set_selpar_L51_HB1(1,7);
init_vector set_selpar_slope1_HB1(1,7);
init_vector set_selpar_L52_HB1(1,7);
init_vector set_selpar_slope2_HB1(1,7);

init_vector set_selpar_L50_HB2(1,7);
init_vector set_selpar_slope_HB2(1,7);
init_vector set_selpar_afull_HB2(1,7);
init_vector set_selpar_sigma_HB2(1,7)

// init_vector set_selpar_L50_HB3(1,7); //if using selectivity blocks
// init_vector set_selpar_slope_HB3(1,7); //if using selectivity blocks

// Headboat discard selectivity
init_vector set_selpar_L50_HB_D(1,7);
init_vector set_selpar_slope_HB_D(1,7);
init_vector set_selpar_afull_HB_D(1,7);
init_vector set_selpar_sigma_HB_D(1,7);

// GR selectivity
init_vector set_selpar_L51_GR(1,7);
init_vector set_selpar_slope1_GR(1,7);
init_vector set_selpar_L52_GR(1,7);
init_vector set_selpar_slope2_GR(1,7);

//--index catchability-----------------------------------------------------------------------------
```

```
init_vector set_log_q_cH(1,7);      //catchability coefficient (log) for comm handline index
init_vector set_log_q_mcvt(1,7);    //catchability coefficient (log) for SERFS chevron trap index
//init_vector set_log_q_vid(1,7);      //catchability coefficient (log) for SEFIS video 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_HB(1,7);
init_vector set_log_avg_F_GR(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_HB(1,3);
init_vector set_log_F_dev_GR(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_vector set_log_F_dev_cH_vals(styr_cH_L,endyr_cH_L);
init_vector set_log_F_dev_HB_vals(styr_HB_L,endyr_HB_L);
init_vector set_log_F_dev_GR_vals(styr_GR_L,endyr_GR_L);
```

```
init_vector set_log_F_dev_HB_D_vals(styr_HB_D,endyr_HB_D);

init_vector set_log_F_dev_GR_D_vals(styr_GR_D,endyr_GR_D);

init_vector set_log_rec_dev_vals(styr_rec_dev,endyr_rec_dev);

init_vector set_log_Nage_dev_vals(2,nages);


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

//-- BAM DATA_SECTION: likelihood weights section

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


init_number set_w_L;           //weight for landings

init_number set_w_D;           //weight for discards

init_number set_w_I_cH;         //weight for comm handline index

init_number set_w_I_mcvt;       //weight for SERFS chevron trap index

//init_number set_w_I_vid;        //weight for SEFIS video index

init_number set_w_I_HB;         //weight for headboat index

init_number set_w_I_GR;         //weight for MRFSS index

init_number set_w_lc_cH;        //weight for comm handline len comps

init_number set_w_lc_mcvt;      //weight for SERFS chevron trap len comps

init_number set_w_lc_HB;        //weight for headboat len comps

init_number set_w_lc_HB_D;      //weight for headboat discards len comps

init_number set_w_lc_GR;        //weight for GR (mrip) len comps

init_number set_w_ac_cH;        //weight for comm handline age comps

init_number set_w_ac_mcvt;      //weight for SERFS chevron trap age comps

init_number set_w_ac_HB;        //weight for headboat age comps

//init_number set_w_ac_GR;        //weight for GR (MRIP) age comps

init_number set_w_Nage_init;    //for fitting initial abundance at age (excluding first age)

init_number set_w_rec;          //for fitting S-R curve

init_number set_w_rec_early;    //additional constraint on early years recruitment
```

init_number set_w_rec_end; //additional constraint on ending years recruitment

init_number set_w_fullF; //penalty for any Fapex>3(removed in final phase of optimization)

init_number set_w_Ftune; //weight applied to tuning F (removed in final phase of optimization)

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

//-- BAM DATA_SECTION: miscellaneous stuff section

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

// FL(mm)-weight(whole weight in kg) relationship: W=aL^b

init_number wgtpar_a;

init_number wgtpar_b;

// Weight(whole weight)- fecundity (units=kg) relationship: log(y)=a+bW

init_number fecpar_a;

init_number fecpar_b;

//init_number fecpar_batches; //used if assuming age-invariant batch number

init_vector fecpar_batches(1,nages); //number of annual batches; may be important if age-dependent, otherwise just a scalar

init_number fecpar_scale; //used for scaling annual egg production (10^X eggs)

// Maturity and proportion female at age

init_vector maturity_f_obs(1,nages); //proportion females mature at age

init_vector prop_f_obs(1,nages); //proportion female at age

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

// Natural mortality

init_vector set_M(1,nages); //age-dependent: used in model

```
init_number max_obs_age;        //max observed age, used to scale M, if estimated

// Discard mortality constants
init_number set_Dmort_HB;
init_number set_Dmort_GR;

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

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

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

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

// Threshold sample sizes for length comps
init_number minSS_cH_lenc;
init_number minSS_mcvt_lenc;
```

```
init_number minSS_HB_lenc;
init_number minSS_HB_D_lenc;
init_number minSS_GR_lenc;

//Threshold sample sizes for age comps
init_number minSS_cH_agec;
init_number minSS_mcvt_agec;
init_number minSS_HB_agec;
//init_number minSS_GR_agec;

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

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

number sqrt2pi;
number g2mt;                //conversion of grams to metric tons
number g2kg;                //conversion of grams to kg
number g2klb;               //conversion of grams to 1000 lb
number mt2klb;               //conversion of metric tons to 1000 lb
number mt2lb;               //conversion of metric tons to lb
number dzero;               //small additive constant to prevent division by zero
number huge_number;         //huge number, to avoid irregular parameter space

init_number use_landings_growth;
```

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



PARAMETER_SECTION


 LOCAL_CALCS
  //POPULATION growth
  const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
  const double K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
  const double t0_LO=set_t0(2); const double t0_HI=set_t0(3); const double t0_PH=set_t0(4);
  const double len_cv_LO=set_len_cv(2); const double len_cv_HI=set_len_cv(3); const double
len_cv_PH=set_len_cv(4);


  //FISHERY LANDINGS growth
  const double Linf_L_LO=set_Linf_L(2); const double Linf_L_HI=set_Linf_L(3); const double
Linf_L_PH=set_Linf_L(4);
  const double K_L_LO=set_K_L(2); const double K_L_HI=set_K_L(3); const double K_L_PH=set_K_L(4);
  const double t0_L_LO=set_t0_L(2); const double t0_L_HI=set_t0_L(3); const double t0_L_PH=set_t0_L(4);
```

```
  const double len_cv_L_LO=set_len_cv_L(2); const double len_cv_L_HI=set_len_cv_L(3); const double
len_cv_L_PH=set_len_cv_L(4);


  //FISHERY INDEPENDENT growth (SERFS chevron trap)

  const double Linf_mcvt_LO=set_Linf_mcvt(2); const double Linf_mcvt_HI=set_Linf_mcvt(3); const double
Linf_mcvt_PH=set_Linf_mcvt(4);

  const double K_mcvt_LO=set_K_mcvt(2); const double K_mcvt_HI=set_K_mcvt(3); const double
K_mcvt_PH=set_K_mcvt(4);

  const double t0_mcvt_LO=set_t0_mcvt(2); const double t0_mcvt_HI=set_t0_mcvt(3); const double
t0_mcvt_PH=set_t0_mcvt(4);

  const double len_cv_mcvt_LO=set_len_cv_mcvt(2); const double len_cv_mcvt_HI=set_len_cv_mcvt(3); const
double len_cv_mcvt_PH=set_len_cv_mcvt(4);


  const double M_constant_LO=set_M_constant(2); const double M_constant_HI=set_M_constant(3); const double
M_constant_PH=set_M_constant(4);


  const double steep_LO=set_steep(2); const double steep_HI=set_steep(3); const double steep_PH=set_steep(4);

  const double log_R0_LO=set_log_R0(2); const double log_R0_HI=set_log_R0(3); const double
log_R0_PH=set_log_R0(4);

  const double R_autocorr_LO=set_R_autocorr(2); const double R_autocorr_HI=set_R_autocorr(3); const double
R_autocorr_PH=set_R_autocorr(4);

  const double rec_sigma_LO=set_rec_sigma(2); const double rec_sigma_HI=set_rec_sigma(3); const double
rec_sigma_PH=set_rec_sigma(4);


  const double selpar_L50_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 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_cH2_LO=set_selpar_L50_cH2(2); const double
selpar_L50_cH2_HI=set_selpar_L50_cH2(3); const double selpar_L50_cH2_PH=set_selpar_L50_cH2(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_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_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_mcvt_LO=set_selpar_L50_mcvt(2); const double
selpar_L50_mcvt_HI=set_selpar_L50_mcvt(3); const double selpar_L50_mcvt_PH=set_selpar_L50_mcvt(4);

  const double selpar_slope_mcvt_LO=set_selpar_slope_mcvt(2); const double
selpar_slope_mcvt_HI=set_selpar_slope_mcvt(3); const double selpar_slope_mcvt_PH=set_selpar_slope_mcvt(4);



  const double selpar_L51_HB1_LO=set_selpar_L51_HB1(2); const double
selpar_L51_HB1_HI=set_selpar_L51_HB1(3); const double selpar_L51_HB1_PH=set_selpar_L51_HB1(4);

  const double selpar_slope1_HB1_LO=set_selpar_slope1_HB1(2); const double
selpar_slope1_HB1_HI=set_selpar_slope1_HB1(3); const double
selpar_slope1_HB1_PH=set_selpar_slope1_HB1(4);

  const double selpar_L52_HB1_LO=set_selpar_L52_HB1(2); const double
selpar_L52_HB1_HI=set_selpar_L52_HB1(3); const double selpar_L52_HB1_PH=set_selpar_L52_HB1(4);

  const double selpar_slope2_HB1_LO=set_selpar_slope2_HB1(2); const double
selpar_slope2_HB1_HI=set_selpar_slope2_HB1(3); const double
selpar_slope2_HB1_PH=set_selpar_slope2_HB1(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_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_afull_HB2_LO=set_selpar_afull_HB2(2); const double
selpar_afull_HB2_HI=set_selpar_afull_HB2(3); const double selpar_afull_HB2_PH=set_selpar_afull_HB2(4);

  const double selpar_sigma_HB2_LO=set_selpar_sigma_HB2(2); const double
selpar_sigma_HB2_HI=set_selpar_sigma_HB2(3); const double
selpar_sigma_HB2_PH=set_selpar_sigma_HB2(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_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_HB_D_LO=set_selpar_L50_HB_D(2); const double
selpar_L50_HB_D_HI=set_selpar_L50_HB_D(3); const double selpar_L50_HB_D_PH=set_selpar_L50_HB_D(4);

  const double selpar_slope_HB_D_LO=set_selpar_slope_HB_D(2); const double
selpar_slope_HB_D_HI=set_selpar_slope_HB_D(3); const double
selpar_slope_HB_D_PH=set_selpar_slope_HB_D(4);
```

```
  const double selpar_afull_HB_D_LO=set_selpar_afull_HB_D(2); const double
selpar_afull_HB_D_HI=set_selpar_afull_HB_D(3); const double
selpar_afull_HB_D_PH=set_selpar_afull_HB_D(4);

  const double selpar_sigma_HB_D_LO=set_selpar_sigma_HB_D(2); const double
selpar_sigma_HB_D_HI=set_selpar_sigma_HB_D(3); const double
selpar_sigma_HB_D_PH=set_selpar_sigma_HB_D(4);



  const double selpar_L51_GR_LO=set_selpar_L51_GR(2); const double
selpar_L51_GR_HI=set_selpar_L51_GR(3); const double selpar_L51_GR_PH=set_selpar_L51_GR(4);

  const double selpar_slope1_GR_LO=set_selpar_slope1_GR(2); const double
selpar_slope1_GR_HI=set_selpar_slope1_GR(3); const double selpar_slope1_GR_PH=set_selpar_slope1_GR(4);

  const double selpar_L52_GR_LO=set_selpar_L52_GR(2); const double
selpar_L52_GR_HI=set_selpar_L52_GR(3); const double selpar_L52_GR_PH=set_selpar_L52_GR(4);

  const double selpar_slope2_GR_LO=set_selpar_slope2_GR(2); const double
selpar_slope2_GR_HI=set_selpar_slope2_GR(3); const double selpar_slope2_GR_PH=set_selpar_slope2_GR(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 log_q_mcvt_LO=set_log_q_mcvt(2); const double log_q_mcvt_HI=set_log_q_mcvt(3); const double
log_q_mcvt_PH=set_log_q_mcvt(4);

//const double log_q_vid_LO=set_log_q_vid(2); const double log_q_vid_HI=set_log_q_vid(3); const double
log_q_vid_PH=set_log_q_vid(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_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 F_init_LO=set_F_init(2); const double F_init_HI=set_F_init(3); const double
F_init_PH=set_F_init(4);

  const double log_avg_F_cH_LO=set_log_avg_F_cH(2); const double log_avg_F_cH_HI=set_log_avg_F_cH(3);
const double log_avg_F_cH_PH=set_log_avg_F_cH(4);

  const double log_avg_F_HB_LO=set_log_avg_F_HB(2); const double log_avg_F_HB_HI=set_log_avg_F_HB(3);
const double log_avg_F_HB_PH=set_log_avg_F_HB(4);

  const double log_avg_F_GR_LO=set_log_avg_F_GR(2); const double log_avg_F_GR_HI=set_log_avg_F_GR(3);
const double log_avg_F_GR_PH=set_log_avg_F_GR(4);

  const double log_avg_F_HB_D_LO=set_log_avg_F_HB_D(2); const double
log_avg_F_HB_D_HI=set_log_avg_F_HB_D(3); const double log_avg_F_HB_D_PH=set_log_avg_F_HB_D(4);
```

```
  const double log_avg_F_GR_D_LO=set_log_avg_F_GR_D(2); const double
log_avg_F_GR_D_HI=set_log_avg_F_GR_D(3); const double log_avg_F_GR_D_PH=set_log_avg_F_GR_D(4);


  //-dev vectors-----------------------------------------------------------------------------------------------------

  const double log_F_dev_cH_LO=set_log_F_dev_cH(1); const double log_F_dev_cH_HI=set_log_F_dev_cH(2);
const double log_F_dev_cH_PH=set_log_F_dev_cH(3);

  const double log_F_dev_HB_LO=set_log_F_dev_HB(1); const double log_F_dev_HB_HI=set_log_F_dev_HB(2);
const double log_F_dev_HB_PH=set_log_F_dev_HB(3);

  const double log_F_dev_GR_LO=set_log_F_dev_GR(1); const double log_F_dev_GR_HI=set_log_F_dev_GR(2);
const double log_F_dev_GR_PH=set_log_F_dev_GR(3);


  const double log_F_dev_HB_D_LO=set_log_F_dev_HB_D(1); const double
log_F_dev_HB_D_HI=set_log_F_dev_HB_D(2); const double log_F_dev_HB_D_PH=set_log_F_dev_HB_D(3);

  const double log_F_dev_GR_D_LO=set_log_F_dev_GR_D(1); const double
log_F_dev_GR_D_HI=set_log_F_dev_GR_D(2); const double log_F_dev_GR_D_PH=set_log_F_dev_GR_D(3);


  const double log_rec_dev_LO=set_log_rec_dev(1); const double log_rec_dev_HI=set_log_rec_dev(2); const
double log_rec_dev_PH=set_log_rec_dev(3);

  const double log_Nage_dev_LO=set_log_Nage_dev(1); const double log_Nage_dev_HI=set_log_Nage_dev(2);
const double log_Nage_dev_PH=set_log_Nage_dev(3);


 END_CALCS


////--------------Growth-----------------------------------------------------------------------------

  //POPULATION GROWTH CURVE

  init_bounded_number Linf(Linf_LO,Linf_HI,Linf_PH);

  init_bounded_number K(K_LO,K_HI,K_PH);

  init_bounded_number t0(t0_LO,t0_HI,t0_PH);

  init_bounded_number len_cv_val(len_cv_LO,len_cv_HI,len_cv_PH);


  //FISHERY LANDINGS GROWTH CURVE

  init_bounded_number Linf_L(Linf_L_LO,Linf_L_HI,Linf_L_PH);

  init_bounded_number K_L(K_L_LO,K_L_HI,K_L_PH);
```

```
init_bounded_number t0_L(t0_L_LO,t0_L_HI,t0_L_PH);
init_bounded_number len_cv_val_L(len_cv_L_LO,len_cv_L_HI,len_cv_L_PH);

//FISHERY INDEPENDENT (SERFS chevron trap) GROWTH CURVE
init_bounded_number Linf_mcvt(Linf_mcvt_LO,Linf_mcvt_HI,Linf_mcvt_PH);
init_bounded_number K_mcvt(K_mcvt_LO,K_mcvt_HI,K_mcvt_PH);
init_bounded_number t0_mcvt(t0_mcvt_LO,t0_mcvt_HI,t0_mcvt_PH);
init_bounded_number len_cv_val_mcvt(len_cv_mcvt_LO,len_cv_mcvt_HI,len_cv_mcvt_PH);

vector Linf_out(1,8);
vector K_out(1,8);
vector t0_out(1,8);
vector len_cv_val_out(1,8);

vector Linf_L_out(1,8);
vector K_L_out(1,8);
vector t0_L_out(1,8);
vector len_cv_val_L_out(1,8);

vector Linf_mcvt_out(1,8);
vector K_mcvt_out(1,8);
vector t0_mcvt_out(1,8);
vector len_cv_val_mcvt_out(1,8);

//POPULATION GROWTH CURVE
vector meanlen_FL(1,nages);   //mean fork length (mm) at age all fish
vector wgt_g(1,nages);        //whole wgt in g
vector wgt_kg(1,nages);       //whole wgt in kg
vector wgt_mt(1,nages);       //whole wgt in mt
```

```
 vector wgt_klb(1,nages);      //whole wgt in 1000 lb
 vector wgt_lb(1,nages);       //whole wgt in lb
 vector fecundity(1,nages);  //KC added

 //These  values used for intermediate and summary calcs as needed
 //FISHERY LANDINGS GROWTH CURVE
 vector meanlen_FL_L(1,nages);   //mean fork length (mm) at age all fish
 vector wgt_g_L(1,nages);        //whole wgt in g
 vector wgt_kg_L(1,nages);       //whole wgt in kg
 vector wgt_mt_L(1,nages);       //whole wgt in mt
 vector wgt_klb_L(1,nages);      //whole wgt in 1000 lb
 vector wgt_lb_L(1,nages);       //whole wgt in lb

 //FISHERY INDEPENDENT (SERFS chevron trap) GROWTH CURVE
 vector meanlen_FL_mcvt(1,nages);   //mean fork length (mm) at age all fish

 matrix len_cH_mm(styr,endyr,1,nages);          //mean length at age of commercial handline landings in mm (may
differ from popn mean)
 matrix wholewgt_cH_klb(styr,endyr,1,nages);    //whole wgt of commercial handline landings in 1000 lb
 matrix len_mcvt_mm(styr,endyr,1,nages);        //mean length at age of SERFS chevron trap in mm (may differ
from popn mean)
 matrix wholewgt_mcvt_klb(styr,endyr,1,nages);  //whole wgt of SERFS chevron trap in 1000 lb
 matrix len_HB_mm(styr,endyr,1,nages);          //mean length at age of HB landings in mm (may differ from popn
mean)
 matrix wholewgt_HB_klb(styr,endyr,1,nages);    //whole wgt of HB landings in 1000 lb
 matrix len_GR_mm(styr,endyr,1,nages);          //mean length at age of GR landings in mm (may differ from popn
mean)
 matrix wholewgt_GR_klb(styr,endyr,1,nages);    //whole wgt of GR landings in 1000 lb
 matrix len_HB_D_mm(styr,endyr,1,nages);        //mean length at age of HB discards in mm (may differ from popn
mean)
 matrix wholewgt_HB_D_klb(styr,endyr,1,nages);  //whole wgt of HB discards in 1000 lb
```

```
  matrix len_GR_D_mm(styr,endyr,1,nages);        //mean length at age of GR discards in mm (may differ from popn
mean)
  matrix wholewgt_GR_D_klb(styr,endyr,1,nages);  //whole wgt of GR discards in 1000 lb
  matrix lenprob(1,nages,1,nlenbins);            //distn of size at age (age-length key, 3 cm bins) in population
  matrix lenprob2(1,nages,1,nlenbins);           //distn of size at age (age-length key, 3 cm bins) in population
  number zscore_len;                            //standardized normal values used for computing lenprob
  number zscore_len2;                           //standardized normal values used for computing lenprob
  vector cprob_lenvec(1,nlenbins);              //cumulative probabilities used for computing lenprob
  vector cprob_lenvec2(1,nlenbins);             //cumulative probabilities used for computing lenprob
  number zscore_lzero;                          //standardized normal values for length = 0
  number cprob_lzero;                           //length probability mass below zero, used for computing lenprob

  number zscore_lzero2;                         //standardized normal values for length = 0
  number cprob_lzero2;                          //length probability mass below zero, used for computing lenprob

  //matrices below are used to match length comps
  matrix lenprob_cH(1,nages,1,nlenbins);     //distn of size at age in cH
  matrix lenprob_mcvt(1,nages,1,nlenbins);   //distn of size at age in SERFS chevron trap
  matrix lenprob_HB(1,nages,1,nlenbins);     //distn of size at age in HB
  matrix lenprob_HB_D(1,nages,1,nlenbins);   //distn of size at age in HB discards
  matrix lenprob_GR(1,nages,1,nlenbins);     //distn of size at age in GR

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

  //LANDINGS
  vector len_sd_L(1,nages);
```

```
  vector len_cv_L(1,nages); //for fishgraph

  //SERFS chevron trap
  vector len_sd_mcvt(1,nages);
  vector len_cv_mcvt(1,nages); //for fishgraph

//----Predicted length and age compositions
  matrix pred_cH_lenc(1,nyr_cH_lenc,1,nlenbins);
  matrix pred_mcvt_lenc(1,nyr_mcvt_lenc,1,nlenbins);
  matrix pred_HB_lenc(1,nyr_HB_lenc,1,nlenbins);
  matrix pred_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);
  matrix pred_GR_lenc(1,nyr_GR_lenc,1,nlenbins);

  matrix pred_cH_agec(1,nyr_cH_agec,1,nages_agec);
  matrix pred_cH_agec_allages(1,nyr_cH_agec,1,nages);
  matrix ErrorFree_cH_agec(1,nyr_cH_agec,1,nages);
  matrix pred_mcvt_agec(1,nyr_mcvt_agec,1,nages_agec);
  matrix pred_mcvt_agec_allages(1,nyr_mcvt_agec,1,nages);
  matrix ErrorFree_mcvt_agec(1,nyr_mcvt_agec,1,nages);
  matrix pred_HB_agec(1,nyr_HB_agec,1,nages_agec);
  matrix pred_HB_agec_allages(1,nyr_HB_agec,1,nages);
  matrix ErrorFree_HB_agec(1,nyr_HB_agec,1,nages);
  //matrix pred_GR_agec(1,nyr_GR_agec,1,nages_agec);
  //matrix pred_GR_agec_allages(1,nyr_GR_agec,1,nages);
  //matrix ErrorFree_GR_agec(1,nyr_GR_agec,1,nages);

//Effective sample size applied in multinomial distributions
  vector nsamp_cH_lenc_allyr(styr,endyr);
  vector nsamp_mcvt_lenc_allyr(styr,endyr);
```

```
  vector nsamp_HB_lenc_allyr(styr,endyr);
  vector nsamp_HB_D_lenc_allyr(styr,endyr);
  vector nsamp_GR_lenc_allyr(styr,endyr);
  vector nsamp_cH_agec_allyr(styr,endyr);
  vector nsamp_mcvt_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);
  //vector nsamp_GR_agec_allyr(styr,endyr);

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_cH_lenc_allyr(styr,endyr);
  vector nfish_mcvt_lenc_allyr(styr,endyr);
  vector nfish_HB_lenc_allyr(styr,endyr);
  vector nfish_HB_D_lenc_allyr(styr,endyr);
  vector nfish_GR_lenc_allyr(styr,endyr);
  vector nfish_cH_agec_allyr(styr,endyr);
  vector nfish_mcvt_agec_allyr(styr,endyr);
  vector nfish_HB_agec_allyr(styr,endyr);
  //vector nfish_GR_agec_allyr(styr,endyr);

//Computed effective sample size for output (not used in fitting)
  vector neff_cH_lenc_allyr_out(styr,endyr);
  vector neff_mcvt_lenc_allyr_out(styr,endyr);
  vector neff_HB_lenc_allyr_out(styr,endyr);
  vector neff_GR_lenc_allyr_out(styr,endyr);
  vector neff_HB_D_lenc_allyr_out(styr,endyr);
  vector neff_cH_agec_allyr_out(styr,endyr);
  vector neff_mcvt_agec_allyr_out(styr,endyr);
  vector neff_HB_agec_allyr_out(styr,endyr);
  //vector neff_GR_agec_allyr_out(styr,endyr);
```

```
//-----Population---------------------------------------------------------------------------------

  matrix N(styr,endyr+1,1,nages);          //Population numbers by year and age at start of yr

  matrix N_mdyr(styr,endyr,1,nages);       //Population numbers by year and age at mdpt of yr: used for comps and
cpue

  matrix N_spawn(styr,endyr,1,nages);     //Population numbers by year and age at peaking spawning: used for SSB
(proj yr ok bc of ssb on Jan1)

  init_bounded_vector log_Nage_dev(2,nages,log_Nage_dev_LO,log_Nage_dev_HI,log_Nage_dev_PH);

  vector log_Nage_dev_output(1,nages);      //used in output. equals zero for first age

  matrix B(styr,endyr+1,1,nages);          //Population biomass by year and age at start of yr

  vector totB(styr,endyr+1);               //Total biomass by year

  vector totN(styr,endyr+1);               //Total abundance by year

  vector SSB(styr,endyr);                 //Total spawning biomass by year (female + male mature biomass) (proj yr ok bc
of ssb on Jan1)

  vector MatFemB(styr,endyr);            //Total spawning biomass by year (mature female biomass) (proj yr ok bc of
ssb on Jan1)

  vector rec(styr,endyr+1);               //Recruits by year

  vector prop_f(1,nages)

  vector maturity_f(1,nages);             //Proportion of female mature at age

  vector reprod(1,nages);                 //vector used to compute spawning biomass (total mature female biomass)

  vector reprod2(1,nages);


//---Stock-Recruit Function (Beverton-Holt, steepness parameterization)----------------------------------

  init_bounded_number log_R0(log_R0_LO,log_R0_HI,log_R0_PH);  //log(virgin Recruitment)

  vector log_R0_out(1,8);

  number R0;                                    //virgin recruitment

  init_bounded_number steep(steep_LO,steep_HI,steep_PH);      //steepness

  vector steep_out(1,8);

  init_bounded_number rec_sigma(rec_sigma_LO,rec_sigma_HI,rec_sigma_PH);  //sd recruitment residuals

  vector rec_sigma_out(1,8);

  init_bounded_number R_autocorr(R_autocorr_LO,R_autocorr_HI,R_autocorr_PH);  //autocorrelation in SR
```

```
  vector R_autocorr_out(1,8);


  number rec_sigma_sq;                    //square of rec_sigma

  number rec_logL_add;                    //additive term in -logL term


  init_bounded_dev_vector
log_rec_dev(styr_rec_dev,endyr_rec_dev,log_rec_dev_LO,log_rec_dev_HI,log_rec_dev_PH);

  vector log_rec_dev_output(styr,endyr+1);            //used in t.series output. equals zero except for yrs in
log_rec_dev

  vector log_rec_dev_out(styr_rec_dev,endyr_rec_dev);  //used in output for bound checking


  number var_rec_dev;                         //variance of log recruitment deviations, from yrs with unconstrainted S-
R(XXXX-XXXX)

  number sigma_rec_dev;                       //sample SD of log residuals (may not equal rec_sigma


  number BiasCor;                         //Bias correction in equilibrium recruits

  number S0;                             //equal to spr_F0*R0 = virgin SSB

  number B0;                             //equal to bpr_F0*R0 = virgin B

  number R1;                             //Recruits in styr

  number R_virgin;                        //unfished recruitment with bias correction

  vector SdS0(styr,endyr);                //SSB / virgin SSB (projection yr possible bc of SSB on Jan1


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

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


//Commercial handline-------------------------------------------------

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

  init_bounded_number selpar_L50_cH1(selpar_L50_cH1_LO,selpar_L50_cH1_HI,selpar_L50_cH1_PH);

  init_bounded_number selpar_slope_cH1(selpar_slope_cH1_LO,selpar_slope_cH1_HI,selpar_slope_cH1_PH);

//  init_bounded_number selpar_L50_cH2(selpar_L50_cH2_LO,selpar_L50_cH2_HI,selpar_L50_cH2_PH);
```

```
//  init_bounded_number selpar_slope_cH2(selpar_slope_cH2_LO,selpar_slope_cH2_HI,selpar_slope_cH2_PH);
//  init_bounded_number selpar_L50_cH3(selpar_L50_cH3_LO,selpar_L50_cH3_HI,selpar_L50_cH3_PH);
//  init_bounded_number selpar_slope_cH3(selpar_slope_cH3_LO,selpar_slope_cH3_HI,selpar_slope_cH3_PH);
  vector selpar_L50_cH1_out(1,8);
  vector selpar_slope_cH1_out(1,8);
//  vector selpar_L50_cH2_out(1,8);
//  vector selpar_slope_cH2_out(1,8);
//  vector selpar_L50_cH3_out(1,8);
//  vector selpar_slope_cH3_out(1,8);

//SERFS chevron trap----------------------------------------------------
  matrix sel_mcvt(styr,endyr,1,nages);
  init_bounded_number selpar_L50_mcvt(selpar_L50_mcvt_LO,selpar_L50_mcvt_HI,selpar_L50_mcvt_PH);
  init_bounded_number selpar_slope_mcvt(selpar_slope_mcvt_LO,selpar_slope_mcvt_HI,selpar_slope_mcvt_PH);
  vector selpar_L50_mcvt_out(1,8);
  vector selpar_slope_mcvt_out(1,8);

//SEFIS video-----------------------------------------------------------
//  matrix sel_vid(styr,endyr,1,nages);
//  vector selpar_L50_vid_out(1,8);
//  vector selpar_slope_vid_out(1,8);

//Recreational (HB)-----------------------------------------------------
  matrix sel_HB(styr,endyr,1,nages);
  init_bounded_number selpar_L51_HB1(selpar_L51_HB1_LO,selpar_L51_HB1_HI,selpar_L51_HB1_PH);
  init_bounded_number
selpar_slope1_HB1(selpar_slope1_HB1_LO,selpar_slope1_HB1_HI,selpar_slope1_HB1_PH);
  init_bounded_number selpar_L52_HB1(selpar_L52_HB1_LO,selpar_L52_HB1_HI,selpar_L52_HB1_PH);
  init_bounded_number
selpar_slope2_HB1(selpar_slope2_HB1_LO,selpar_slope2_HB1_HI,selpar_slope2_HB1_PH);
```

```
  init_bounded_number selpar_L50_HB2(selpar_L50_HB2_LO,selpar_L50_HB2_HI,selpar_L50_HB2_PH);
  init_bounded_number selpar_slope_HB2(selpar_slope_HB2_LO,selpar_slope_HB2_HI,selpar_slope_HB2_PH);
  init_bounded_number selpar_afull_HB2(selpar_afull_HB2_LO,selpar_afull_HB2_HI,selpar_afull_HB2_PH);
  init_bounded_number selpar_sigma_HB2(selpar_sigma_HB2_LO,selpar_sigma_HB2_HI,selpar_sigma_HB2_PH)

//  init_bounded_number selpar_L50_HB3(selpar_L50_HB3_LO,selpar_L50_HB3_HI,selpar_L50_HB3_PH);
//  init_bounded_number selpar_slope_HB3(selpar_slope_HB3_LO,selpar_slope_HB3_HI,selpar_slope_HB3_PH);

  vector selpar_L51_HB1_out(1,8);
  vector selpar_slope1_HB1_out(1,8);
  vector selpar_L52_HB1_out(1,8);
  vector selpar_slope2_HB1_out(1,8);

  vector selpar_L50_HB2_out(1,8);
  vector selpar_slope_HB2_out(1,8);
  vector selpar_afull_HB2_out(1,8);
  vector selpar_sigma_HB2_out(1,8);

//  vector selpar_L50_HB3_out(1,8);
//  vector selpar_slope_HB3_out(1,8);

//HB discard selectivities
  matrix sel_HB_D(styr,endyr,1,nages);
  init_bounded_number selpar_L50_HB_D(selpar_L50_HB_D_LO,selpar_L50_HB_D_HI,selpar_L50_HB_D_PH);
  init_bounded_number
selpar_slope_HB_D(selpar_slope_HB_D_LO,selpar_slope_HB_D_HI,selpar_slope_HB_D_PH);
  init_bounded_number
selpar_afull_HB_D(selpar_afull_HB_D_LO,selpar_afull_HB_D_HI,selpar_afull_HB_D_PH);
```

```
  init_bounded_number
selpar_sigma_HB_D(selpar_sigma_HB_D_LO,selpar_sigma_HB_D_HI,selpar_sigma_HB_D_PH);


  vector selpar_L50_HB_D_out(1,8);
  vector selpar_slope_HB_D_out(1,8);
  vector selpar_afull_HB_D_out(1,8);
  vector selpar_sigma_HB_D_out(1,8);


//Recreational (GR)----------------------------------------
  matrix sel_GR(styr,endyr,1,nages);
  init_bounded_number selpar_L51_GR(selpar_L51_GR_LO,selpar_L51_GR_HI,selpar_L51_GR_PH);
  init_bounded_number selpar_slope1_GR(selpar_slope1_GR_LO,selpar_slope1_GR_HI,selpar_slope1_GR_PH);
  init_bounded_number selpar_L52_GR(selpar_L52_GR_LO,selpar_L52_GR_HI,selpar_L52_GR_PH);
  init_bounded_number selpar_slope2_GR(selpar_slope2_GR_LO,selpar_slope2_GR_HI,selpar_slope2_GR_PH);


  vector selpar_L51_GR_out(1,8);
  vector selpar_slope1_GR_out(1,8);
  vector selpar_L52_GR_out(1,8);
  vector selpar_slope2_GR_out(1,8);


//GR discard selectivity
  matrix sel_GR_D(styr,endyr,1,nages);
//  vector selpar_L50_GR_D_out(1,8); //set to HB_D selectivity
//  vector selpar_slope_GR_D_out(1,8); //set to HB_D selectivity


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

```
  vector sel_wgted_tot(1,nages);//toward Z, landings plus deads discards


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

//-------CPUE Predictions--------------------------------

  vector pred_cH_cpue(styr_cH_cpue,endyr_cH_cpue);        //predicted cH index (weight fish per effort)

  matrix N_cH(styr_cH_cpue,endyr_cH_cpue,1,nages);        //used to compute cH index

  vector pred_mcvt_cpue(styr_mcvt_cpue,endyr_mcvt_cpue);  //predicted SERFS chevron trap  index (weight fish
per effort)

  matrix N_mcvt(styr_mcvt_cpue,endyr_mcvt_cpue,1,nages);  //used to compute SERFS chevron trap index

//  vector pred_vid_cpue(styr_vid_cpue,endyr_vid_cpue);   //predicted video index (weight fish per effort)

//  matrix N_vid(styr_vid_cpue,endyr_vid_cpue,1,nages);   //used to compute video index

  vector pred_HB_cpue(styr_HB_cpue,endyr_HB_cpue);        //predicted HB index (number fish per effort)

  matrix N_HB(styr_HB_cpue,endyr_HB_cpue,1,nages);        //used to compute HB index

  vector pred_GR_cpue(styr_GR_cpue,endyr_GR_cpue);        //predicted GR index (number fish per effort)

  matrix N_GR(styr_GR_cpue,endyr_GR_cpue,1,nages);        //used to compute GR index



//---Catchability (CPUE q's)-----------------------------------------------------------

  init_bounded_number log_q_cH(log_q_cH_LO,log_q_cH_HI,log_q_cH_PH);

  init_bounded_number log_q_mcvt(log_q_mcvt_LO,log_q_mcvt_HI,log_q_mcvt_PH);

//  init_bounded_number log_q_vid(log_q_vid_LO,log_q_vid_HI,log_q_vid_PH);

  init_bounded_number log_q_HB(log_q_HB_LO,log_q_HB_HI,log_q_HB_PH);

  init_bounded_number log_q_GR(log_q_GR_LO,log_q_GR_HI,log_q_GR_PH);

  vector log_q_cH_out(1,8);

  vector log_q_mcvt_out(1,8);

//  vector log_q_vid_out(1,8);

  vector log_q_HB_out(1,8);

  vector log_q_GR_out(1,8);
```

```
//  init_bounded_number q_rate(0.001,0.1,set_q_rate_phase); //not estimated

  number q_rate;

  vector q_rate_fcn_cH(styr_cH_cpue,endyr_cH_cpue);         //increase due to technology creep (saturates in 2003)

  vector q_rate_fcn_mcvt(styr_mcvt_cpue,endyr_mcvt_cpue);   //increase due to technology creep (saturates in
2003)

//  vector q_rate_fcn_vid(styr_vid_cpue,endyr_vid_cpue);    //increase due to technology creep (saturates in 2003)

  vector q_rate_fcn_HB(styr_HB_cpue,endyr_HB_cpue);         //increase due to technology creep (saturates in 2003)

  vector q_rate_fcn_GR(styr_GR_cpue,endyr_GR_cpue);         //increase due to technology creep (saturates in 2003)


//  init_bounded_number q_DD_beta(0.1,0.9,set_q_DD_phase);    //not estimated

  number q_DD_beta;

  vector q_DD_fcn(styr,endyr);    //density dependent function as a multiple of q (scaled a la Katsukawa and
Matsuda. 2003)

  number B0_q_DD;                 //B0 of ages q_DD_age plus

  vector B_q_DD(styr,endyr);      //annual biomass of ages q_DD_age plus


//Fishery dependent random walk catchability

//  init_bounded_vector q_RW_log_dev_HB(styr_HB_cpue,endyr_HB_cpue-1,-3.0,3.0,set_q_RW_phase); //not
estimated in this model

 vector q_RW_log_dev_cH(styr_cH_cpue,endyr_cH_cpue-1);

 vector q_RW_log_dev_mcvt(styr_mcvt_cpue,endyr_mcvt_cpue-1);

// vector q_RW_log_dev_vid(styr_vid_cpue,endyr_vid_cpue-1);

 vector q_RW_log_dev_HB(styr_HB_cpue,endyr_HB_cpue-1);

 vector q_RW_log_dev_GR(styr_GR_cpue,endyr_GR_cpue-1);


//Fishery dependent catchability over time, may be constant

 vector q_cH(styr_cH_cpue,endyr_cH_cpue);

 vector q_mcvt(styr_mcvt_cpue,endyr_mcvt_cpue)

// vector q_vid(styr_vid_cpue,endyr_vid_cpue)

 vector q_HB(styr_HB_cpue,endyr_HB_cpue);
```

```
 vector q_GR(styr_GR_cpue,endyr_GR_cpue);

//-----------------------------------------------------------------------------------------------------------------------------------
-----
//---Landings in numbers (total or 1000 fish) and in wgt (whole klb)-------------------------------------------------
  matrix L_cH_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_cH_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_cH_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_cH_L_klb(styr,endyr);      //yearly landings in 1000 lb whole summed over ages
  //vector obs_cH_L_wbias(styr,endyr);     //yearly landings observed, perhaps adjusted for multiplicitive bias

  matrix L_HB_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_HB_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_HB_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_HB_L_klb(styr,endyr);      //yearly landings in 1000 lb gutted summed over ages

  matrix L_GR_num(styr,endyr,1,nages);   //landings (numbers) at age
  matrix L_GR_klb(styr,endyr,1,nages);   //landings (1000 lb whole weight) at age
  vector pred_GR_L_knum(styr,endyr);     //yearly landings in 1000 fish summed over ages
  vector pred_GR_L_klb(styr,endyr);      //yearly landings in 1000 lb gutted summed over ages

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

//---Discards (number dead fish) ------------------------------------------------
  matrix D_HB_num(styr,endyr,1,nages);            //discards (numbers) at age
  matrix D_HB_klb(styr,endyr,1,nages);            //discards (1000 lb whole) at age
```

```
  vector pred_HB_D_knum(styr_HB_D,endyr_HB_D);     //yearly dead discards summed over ages
  vector obs_HB_D(styr_HB_D,endyr_HB_D);           //observed releases multiplied by discard mortality
  vector pred_HB_D_klb(styr_HB_D,endyr_HB_D);      //yearly dead discards in klb whole wgt summed over ages

  matrix D_GR_num(styr,endyr,1,nages);             //discards (numbers) at age
  matrix D_GR_klb(styr,endyr,1,nages);             //discards (1000 lb whole) at age
  vector pred_GR_D_knum(styr_GR_D,endyr_GR_D);     //yearly dead discards summed over ages
  vector obs_GR_D(styr_GR_D,endyr_GR_D);           //observed releases multiplied by discard mortality
  vector pred_GR_D_klb(styr_GR_D,endyr_GR_D);      //yearly dead discards in klb whole wgt summed over ages

  matrix D_total_num(styr,endyr,1,nages);          //total discards in number at age
  matrix D_total_klb(styr,endyr,1,nages);          //discards in klb whole wgt at age
  vector D_total_knum_yr(styr,endyr);              //total discards in 1000 fish by yr summed over ages
  vector D_total_klb_yr(styr,endyr);               //total discards (klb whole wgt) by yr summed over ages

  number Dmort_HB;
  number Dmort_GR;

////---MSY calcs-------------------------------------------------------------------------
  number F_cH_prop;       //proportion of F_sum attributable to cH, last X=selpar_n_yrs_wgted yrs
  number F_HB_prop;       //proportion of F_sum attributable to HB, last X=selpar_n_yrs_wgted yrs
  number F_GR_prop;       //proportion of F_sum attributable to GR, last X=selpar_n_yrs_wgted yrs
  number F_HB_D_prop;     //proportion of F_sum attributable to HB discards, last X=selpar_n_yrs_wgted yrs
  number F_GR_D_prop;     //proportion of F_sum attributable to GR discards, last X=selpar_n_yrs_wgted yrs

  number F_init_cH_prop;  //proportion of F_init attributable to cH, first X yrs, No discards in initial yrs
  number F_init_HB_prop;  //proportion of F_init attributable to HB, first X yrs
  number F_init_GR_prop;  //proportion of F_init attributable to GR, first X yrs
  number F_init_HB_D_prop;  //proportion of F_init attributable to HB discards, first X yrs
```

```
number F_init_GR_D_prop;  //proportion of F_init attributable to GR discards, first X yrs

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

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

number SSB_msy_out;          //SSB (total mature biomass) at msy
number F_msy_out;           //F at msy
number msy_klb_out;          //max sustainable yield (1000 lb whole wgt)
number msy_knum_out;         //max sustainable yield (1000 fish)
number D_msy_klb_out;        //discards associated with msy (1000 lb whole wgt)
number D_msy_knum_out;       //discards associated with msy (1000 fish)
number B_msy_out;           //total biomass at MSY
number R_msy_out;           //equilibrium recruitment at F=Fmsy
number spr_msy_out;          //spr at F=Fmsy

number F20_dum;                              //intermediate calculation for F20  added 1-19-16 here to...
number F30_dum;                              //intermediate calculation for F30
number F40_dum;                              //intermediate calculation for F40
number F20_out;                  //F20
number F30_out;                  //F30
number F40_out;                  //F40
number SSB_F30_out;
number B_F30_out;
number R_F30_out;
number L_F30_knum_out;
```

```
  number L_F30_klb_out;
  number D_F30_knum_out;
  number D_F30_klb_out;         //here

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

  vector FdF_msy(styr,endyr);
  vector FdF30(styr,endyr);          //added 1-19-16
  vector SdSSB_msy(styr,endyr);    //(proj yr ok bc of ssb on Jan1)
  number SdSSB_msy_end;
  number FdF_msy_end;
  number FdF_msy_end_mean;           //geometric mean of last X yrs
  vector SdSSB_F30(styr,endyr);          //added 1-19-16 here to...
```

```
  vector Sdmsst_F30(styr,endyr);
  number SdSSB_F30_end;
  number Sdmsst_F30_end;
  number FdF30_end_mean;            //geometric mean of last selpar_n_yrs_wgted yrs
  number Fend_mean_temp;              //intermediate calc for geometric mean of last selpar_n_yrs_wgted yrs
  number Fend_mean;                   // here. geometric mean of last selpar_n_yrs_wgted yrs


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

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

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

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

  vector M(1,nages);                  //age-dependent natural mortality
  init_bounded_number M_constant(M_constant_LO,M_constant_HI,M_constant_PH);   //age-independent: used
only for MSST
  vector M_constant_out(1,8);
```

```
  number smsy2msst;                    //scales Smsy to get msst using (1-M). Used only in output.
  number smsy2msst75;                   //scales Smsy to get msst using 75%. Used only in output.

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

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

  init_bounded_number log_avg_F_HB(log_avg_F_HB_LO,log_avg_F_HB_HI,log_avg_F_HB_PH);
  vector log_avg_F_HB_out(1,8);
  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);
  vector log_F_dev_HB_out(styr_HB_L,endyr_HB_L);
  matrix F_HB(styr,endyr,1,nages);
  vector F_HB_out(styr,endyr); //used for intermediate calculations in fcn 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_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_init_HB_D;
  number log_F_dev_end_HB_D;

  init_bounded_number log_avg_F_GR_D(log_avg_F_GR_D_LO,log_avg_F_GR_D_HI,log_avg_F_GR_D_PH);
  vector log_avg_F_GR_D_out(1,8);
  init_bounded_dev_vector
log_F_dev_GR_D(styr_GR_D,endyr_GR_D,log_F_dev_GR_D_LO,log_F_dev_GR_D_HI,log_F_dev_GR_D_PH);
  vector log_F_dev_GR_D_out(styr_GR_D,endyr_GR_D);
  matrix F_GR_D(styr,endyr,1,nages);
  vector F_GR_D_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_init_GR_D;
  number log_F_dev_end_GR_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
//  vector sel_initial(1,nages);       //initial selectivity (a combination of for-hire and commercial selectivities)

//---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 spr_ratio(1,n_iter_spr);    // added 1-19-16  reproductive capacity-per-recruit relative to spr_F0 values
corresponding to F values in F_spr
  vector L_spr(1,n_iter_spr);        //landings(lb whole)-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_mcvt;
  number sdnr_lc_HB;
  number sdnr_lc_HB_D;
  number sdnr_lc_GR;

  number sdnr_ac_cH;
  number sdnr_ac_mcvt;
  number sdnr_ac_HB;
  number sdnr_ac_GR;

  number sdnr_I_cH;
  number sdnr_I_mcvt;
//  number sdnr_I_vid;
  number sdnr_I_HB;
  number sdnr_I_GR;

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

  number w_I_cH;
  number w_I_mcvt;
//  number w_I_vid;
  number w_I_HB;
  number w_I_GR;
```

```
  number w_lc_cH;
  number w_lc_mcvt;
  number w_lc_HB;
  number w_lc_HB_D;
  number w_lc_GR;

  number w_ac_cH;
  number w_ac_mcvt;
  number w_ac_HB;
  //number w_ac_GR;

  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_HB_L;
  number f_GR_L;

  number f_HB_D;
  number f_GR_D;

  number f_cH_cpue;
  number f_mcvt_cpue;
//  number f_vid_cpue;
  number f_HB_cpue;
```

```
  number f_GR_cpue;

  number f_cH_lenc;
  number f_mcvt_lenc;
  number f_HB_lenc;
  number f_HB_D_lenc;
  number f_GR_lenc;

  number f_cH_agec;
  number f_mcvt_agec;
  number f_HB_agec;
  //number f_GR_agec;

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

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

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

```
  number numer;                 //numerator used in some calculations

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



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

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

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

// Set values of fixed parameters or set initial guess of estimated parameters
  Dmort_HB=set_Dmort_HB;
  obs_HB_D=Dmort_HB*obs_HB_released;
```

```
  Dmort_GR=set_Dmort_GR;
  obs_GR_D=Dmort_GR*obs_GR_released;

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

  // FISHERY LANDINGS Growth
  Linf_L=set_Linf_L(1);
  K_L=set_K_L(1);
  t0_L=set_t0_L(1);
  len_cv_val_L=set_len_cv_L(1);

  // FISHERY INDEPENDENT (SERFS chevron trap) Growth
  Linf_mcvt=set_Linf_mcvt(1);
  K_mcvt=set_K_mcvt(1);
  t0_mcvt=set_t0_mcvt(1);
  len_cv_val_mcvt=set_len_cv_mcvt(1);

  M=set_M;
  M_constant=set_M_constant(1);
  smsy2msst=1.0-M_constant;
  smsy2msst75=0.75;
//  for (iage=1;iage<=max_obs_age;iage++){Mscale_ages(iage)=iage;}

  log_R0=set_log_R0(1);
  steep=set_steep(1);
  R_autocorr=set_R_autocorr(1);
```

```
  rec_sigma=set_rec_sigma(1);

  log_q_cH=set_log_q_cH(1);
  log_q_mcvt=set_log_q_mcvt(1);
//  log_q_vid=set_log_q_vid(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_mcvt=1.0;
//  q_rate_fcn_vid=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_mcvt.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

  w_L=set_w_L;
  w_D=set_w_D;

  w_I_cH=set_w_I_cH;
  w_I_mcvt=set_w_I_mcvt;
//  w_I_vid=set_w_I_vid;
  w_I_HB=set_w_I_HB;
  w_I_GR=set_w_I_GR;

  w_lc_cH=set_w_lc_cH;
  w_lc_mcvt=set_w_lc_mcvt;
```

```
 w_lc_HB=set_w_lc_HB;
 w_lc_HB_D=set_w_lc_HB_D;
 w_lc_GR=set_w_lc_GR;

 w_ac_cH=set_w_ac_cH;
 w_ac_mcvt=set_w_ac_mcvt;
 w_ac_HB=set_w_ac_HB;
 //w_ac_GR=set_w_ac_GR;

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

 F_init=set_F_init(1);

 log_avg_F_cH=set_log_avg_F_cH(1);
 log_avg_F_HB=set_log_avg_F_HB(1);
 log_avg_F_GR=set_log_avg_F_GR(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);

 log_F_dev_cH=set_log_F_dev_cH_vals;
 log_F_dev_HB=set_log_F_dev_HB_vals;
 log_F_dev_GR=set_log_F_dev_GR_vals;
 log_F_dev_HB_D=set_log_F_dev_HB_D_vals;
 log_F_dev_GR_D=set_log_F_dev_GR_D_vals;
```

```
  selpar_L50_cH1=set_selpar_L50_cH1(1);
  selpar_slope_cH1=set_selpar_slope_cH1(1);

  selpar_L50_mcvt=set_selpar_L50_mcvt(1);
  selpar_slope_mcvt=set_selpar_slope_mcvt(1);

  selpar_L51_HB1=set_selpar_L51_HB1(1);
  selpar_slope1_HB1=set_selpar_slope1_HB1(1);
  selpar_L52_HB1=set_selpar_L52_HB1(1);
  selpar_slope2_HB1=set_selpar_slope2_HB1(1);

  selpar_L50_HB2=set_selpar_L50_HB2(1);
  selpar_slope_HB2=set_selpar_slope_HB2(1);
  selpar_afull_HB2=set_selpar_afull_HB2(1);
  selpar_sigma_HB2=set_selpar_sigma_HB2(1);

// selpar_L50_HB3=set_selpar_L50_HB3(1);
// selpar_slope_HB3=set_selpar_slope_HB3(1);

  selpar_L50_HB_D=set_selpar_L50_HB_D(1);
  selpar_slope_HB_D=set_selpar_slope_HB_D(1);
  selpar_afull_HB_D=set_selpar_afull_HB_D(1);
  selpar_sigma_HB_D=set_selpar_sigma_HB_D(1);

  selpar_L51_GR=set_selpar_L51_GR(1);
  selpar_slope1_GR=set_selpar_slope1_GR(1);
  selpar_L52_GR=set_selpar_L52_GR(1);
  selpar_slope2_GR=set_selpar_slope2_GR(1);
```

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

 SSB_msy_out=0.0;

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

 maturity_f=maturity_f_obs;
 prop_f=prop_f_obs;

 //lbins=lenbins; //NOT NEEDED

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

     nsamp_cH_lenc_allyr=missing;
     nsamp_mcvt_lenc_allyr=missing;
     nsamp_HB_lenc_allyr=missing;
     nsamp_HB_D_lenc_allyr=missing;
     nsamp_GR_lenc_allyr=missing;
     nsamp_cH_agec_allyr=missing;
```

```
nsamp_mcvt_agec_allyr=missing;
nsamp_HB_agec_allyr=missing;
//nsamp_GR_agec_allyr=missing;

nfish_cH_lenc_allyr=missing;
nfish_mcvt_lenc_allyr=missing;
nfish_HB_lenc_allyr=missing;
nfish_HB_D_lenc_allyr=missing;
nfish_GR_lenc_allyr=missing;
nfish_cH_agec_allyr=missing;
nfish_mcvt_agec_allyr=missing;
nfish_HB_agec_allyr=missing;
//nfish_GR_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_mcvt_lenc; iyear++)
  {if (nsamp_mcvt_lenc(iyear)>=minSS_mcvt_lenc)
    {nsamp_mcvt_lenc_allyr(yrs_mcvt_lenc(iyear))=nsamp_mcvt_lenc(iyear);
     nfish_mcvt_lenc_allyr(yrs_mcvt_lenc(iyear))=nfish_mcvt_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_HB_D_lenc; iyear++)
    {if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
      {nsamp_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=nsamp_HB_D_lenc(iyear);
       nfish_HB_D_lenc_allyr(yrs_HB_D_lenc(iyear))=nfish_HB_D_lenc(iyear);}}

  for (iyear=1; iyear<=nyr_GR_lenc; iyear++)
    {if (nsamp_GR_lenc(iyear)>=minSS_GR_lenc)
      {nsamp_GR_lenc_allyr(yrs_GR_lenc(iyear))=nsamp_GR_lenc(iyear);
       nfish_GR_lenc_allyr(yrs_GR_lenc(iyear))=nfish_GR_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_mcvt_agec; iyear++)
    {if (nsamp_mcvt_agec(iyear)>=minSS_mcvt_agec)
     {nsamp_mcvt_agec_allyr(yrs_mcvt_agec(iyear))=nsamp_mcvt_agec(iyear);
      nfish_mcvt_agec_allyr(yrs_mcvt_agec(iyear))=nfish_mcvt_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);}}

//   for (iyear=1; iyear<=nyr_GR_agec; iyear++)
//     {if (nsamp_GR_agec(iyear)>=minSS_GR_agec)
//      {nsamp_GR_agec_allyr(yrs_GR_agec(iyear))=nsamp_GR_agec(iyear);
//       nfish_GR_agec_allyr(yrs_GR_agec(iyear))=nfish_GR_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_HB.initialize(); L_HB_num.initialize();
  F_GR.initialize(); L_GR_num.initialize();
  F_HB_D.initialize(); D_HB_num.initialize();
  F_GR_D.initialize(); D_GR_num.initialize();

  F_cH_out.initialize();
  F_HB_out.initialize();
  F_GR_out.initialize();
  F_HB_D_out.initialize();
  F_GR_D_out.initialize();


  sel_cH.initialize();
  sel_mcvt.initialize();
//  sel_vid.initialize();
  sel_HB.initialize();
  sel_HB_D.initialize();
  sel_HB.initialize();
```

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


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

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

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

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

 get_length_weight_at_age();
 //cout << "got length, weight, fecundity transitions" << endl;
 get_reprod();
 //cout << "got repro stuff" << endl;
 get_length_at_age_dist();
```

```
 //cout << "got predicted length at age distribution" << endl;
 get_weight_at_age_landings();
 //cout << "got weight at age of landings" << endl;
 get_spr_F0();
 //cout << "got F0 spr" << endl;
 get_selectivity();
 //cout << "got selectivity" << endl;
 get_mortality();
 //cout << "got mortalities" << endl;
 get_bias_corr();
 //cout << "got recruitment bias correction" << endl;
 get_numbers_at_age();
 //cout << "got numbers at age" << endl;
 get_landings_numbers();
 //cout << "got landings in numbers" << endl;
 get_landings_wgt();
 //cout << "got landings in wgt" << endl;
 get_dead_discards();
//   cout << "got dead discards in num and wgt" << endl;
//   cout << pred_GR_D_knum << endl;
 get_catchability_fcns();
 //cout << "got catchability_fcns" << endl;
 get_indices();
 //cout << "got indices" << endl;
 get_length_comps();
 //cout << "got length comps" << endl;
 get_age_comps();
 //cout << "got age comps" << endl;
 evaluate_objective_function();
```

```
 //cout << "objective function calculations complete" << endl;



FUNCTION get_length_weight_at_age


  //compute mean length (mm TL) and weight (whole) at age

    meanlen_FL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));     //total length in mm


    wgt_kg=wgtpar_a*pow(meanlen_FL,wgtpar_b);            //whole wgt in kg

    wgt_g=wgt_kg/g2kg;                                  //convert wgt in kg to weight in g

    wgt_mt=wgt_g*g2mt;                                   //convert weight in g to weight in mt

    wgt_klb=mt2klb*wgt_mt;                               //1000 lb of whole wgt

    wgt_lb=mt2lb*wgt_mt;                                 //lb of whole wgt


    fecundity=(fecpar_a+fecpar_b*meanlen_FL)/fecpar_scale; //annual egg production of a mature female at age in
units of fecpar_scale


  //THESE CALCULATIONS ASSUME THE FISHERY LANDINGS GROWTH CURVE

    meanlen_FL_L=Linf_L*(1.0-mfexp(-K_L*(agebins-t0_L+0.5))); //fork length in mm

    wgt_kg_L=wgtpar_a*pow(meanlen_FL_L,wgtpar_b);            //wgt in kg KC changed from wgt in metric tons
(wgt_mt) because L-W relationship is mm FL and kg whole weight

    wgt_g_L=wgt_kg_L/g2kg;                                  //KC convert wgt in kg from L-W relationship to weight in g

    wgt_mt_L=wgt_g_L*g2mt;                                   //KC convert weight in g to weight in mt

    wgt_klb_L=mt2klb*wgt_mt_L;                               //1000 lb of whole wgt

    wgt_lb_L=mt2lb*wgt_mt_L;                                 //1000 lb of whole wgt


  //THESE CALCULATIONS ASSUME THE FISHERY (SERFS chevron trap) GROWTH CURVE

    meanlen_FL_mcvt=Linf_mcvt*(1.0-mfexp(-K_mcvt*(agebins-t0_mcvt+0.5)));    //fork length in mm
```

```
FUNCTION get_reprod

  //reprod is product of stuff going into reproductive capacity calcs
//  reprod=elem_prod(elem_prod(prop_f,maturity_f),fecundity);
  reprod=elem_prod(elem_prod(elem_prod(prop_f,maturity_f),fecundity),fecpar_batches);
  reprod2=elem_prod(elem_prod(prop_f,maturity_f),wgt_mt);



FUNCTION get_length_at_age_dist

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

dvar_vector length(1,nages);
 dvariable cvlen;

 if (use_landings_growth==1.0)
                                        //LC added to allow for use of Landings Growth Curve
           {
           length=meanlen_FL_L;
           cvlen=len_cv_val_L;
//         sdlen=len_sd_val_L;
       len_sd=len_sd_L;
           }
  if (use_landings_growth==0.0)
           {
           length=meanlen_FL;
           cvlen=len_cv_val;
//     sdlen=len_sd_val;
       len_sd=len_sd;
```

```
            }

  for (iage=1;iage<=nages;iage++)
  {
    len_cv(iage)=cvlen;
    len_sd(iage)=length(iage)*len_cv(iage);
//      len_sd(iage)=sdlen;
//      len_cv(iage)=len_sd(iage)/length(iage);

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

    //first length bin
    zscore_len=((lenbins(1)+0.5*lenbins_width)-length(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)-length(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)-length(iage)) / len_sd(iage);
    lenprob(iage,nlenbins)=1.0-cumd_norm(zscore_len);

    lenprob(iage)=lenprob(iage)/(1.0-cprob_lzero);  //renormalize to account for any prob mass below size=0
```

```
  }
  //fleet and survey specific length probs, all assumed here to equal the popn
  lenprob_cH=lenprob;
//KC  lenprob_mcvt=lenprob;
  lenprob_HB=lenprob;
  lenprob_GR=lenprob;
  lenprob_HB_D=lenprob;

    ////////////// THIS SECTION FOR SERFS chevron trap (mcvt)  ////////////////////////
  lenprob2.initialize();
  length=meanlen_FL_mcvt;
  cvlen=len_cv_val_mcvt;
//  sdlen=len_sd_val_mcvt;
  len_sd=len_sd_mcvt;

  for (iage=1;iage<=nages;iage++)
    {
    len_cv(iage)=cvlen;

    len_sd(iage)=length(iage)*len_cv(iage);

//      len_sd(iage)=sdlen;
//      len_cv(iage)=len_sd(iage)/length(iage);

      zscore_lzero2=(0.0-length(iage))/len_sd(iage);
      cprob_lzero2=cumd_norm(zscore_lzero2);

      zscore_len2=((lenbins(1)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
      cprob_lenvec2(1)=cumd_norm(zscore_len2);
      lenprob2(iage,1)=cprob_lenvec2(1)-cprob_lzero2;
```

```
    //most other length bins
    for (ilen=2;ilen<nlenbins;ilen++)
      {
       zscore_len2=((lenbins(ilen)+0.5*lenbins_width)-length(iage)) / len_sd(iage);
       cprob_lenvec2(ilen)=cumd_norm(zscore_len2);
       lenprob2(iage,ilen)=cprob_lenvec2(ilen)-cprob_lenvec2(ilen-1);
      }
    //last length bin is a plus group
    zscore_len2=((lenbins(nlenbins)-0.5*lenbins_width)-length(iage)) / len_sd(iage);
       lenprob2(iage,nlenbins)=1.0-cumd_norm(zscore_len2);
    lenprob2(iage)=lenprob2(iage)/(1.0-cprob_lzero2);  //renormalize to account for any prob mass below size=0
   }
  //fleet and survey specific length probs
  lenprob_mcvt=lenprob2;

FUNCTION get_weight_at_age_landings

 //KC5--modified based on Lew code
  if(use_landings_growth==1.0){

 for (iyear=styr; iyear<=endyr; iyear++)
 {
  len_cH_mm(iyear)=meanlen_FL_L;
  wholewgt_cH_klb(iyear)=wgt_klb_L; //wholeweight used to match index
  len_HB_mm(iyear)=meanlen_FL_L;
  wholewgt_HB_klb(iyear)=wgt_klb_L; //KC6  change _out to _Lc
  len_GR_mm(iyear)=meanlen_FL_L;
  wholewgt_GR_klb(iyear)=wgt_klb_L; //KC6   change _out to _L
```

```
  len_HB_D_mm(iyear)=meanlen_FL_L;
  wholewgt_HB_D_klb(iyear)=wgt_klb_L;  //KC6; change _out to _L
  len_GR_D_mm(iyear)=meanlen_FL_L;
  wholewgt_GR_D_klb(iyear)=wgt_klb_L;  //KC6  change _out to _L
 }

 }

 if(use_landings_growth==0.0){

 for (iyear=styr; iyear<=endyr; iyear++)
 {
  len_cH_mm(iyear)=meanlen_FL;
  wholewgt_cH_klb(iyear)=wgt_klb; //wholeweight used to match index
  len_HB_mm(iyear)=meanlen_FL;
  wholewgt_HB_klb(iyear)=wgt_klb;  //KC6  KC6-got rid _out
  len_GR_mm(iyear)=meanlen_FL;
  wholewgt_GR_klb(iyear)=wgt_klb;  //KC6  KC6-got rid _out

  len_HB_D_mm(iyear)=meanlen_FL;
  wholewgt_HB_D_klb(iyear)=wgt_klb;  //KC6  KC6-got rid _out
  len_GR_D_mm(iyear)=meanlen_FL;
  wholewgt_GR_D_klb(iyear)=wgt_klb;  //KC6  KC6-got rid _out
 }

 }
```

```
FUNCTION get_spr_F0


  //at mdyr, apply half this yr's mortality, half next yr's

  N_spr_F0(1)=1.0*mfexp(-1.0*M(1)*spawn_time_frac); //at peak spawning time

  N_bpr_F0(1)=1.0;      //at start of year

  for (iage=2; iage<=nages; iage++)

  {

    //N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(M(iage-1));

    N_spr_F0(iage)=N_spr_F0(iage-1)*mfexp(-1.0*(M(iage-1)*(1.0-spawn_time_frac) +
M(iage)*spawn_time_frac));

    N_bpr_F0(iage)=N_bpr_F0(iage-1)*mfexp(-1.0*(M(iage-1)));

  }

  N_spr_F0(nages)=N_spr_F0(nages)/(1.0-mfexp(-1.0*M(nages))); //plus group (sum of geometric series)

  N_bpr_F0(nages)=N_bpr_F0(nages)/(1.0-mfexp(-1.0*M(nages)));


  spr_F0=sum(elem_prod(N_spr_F0,reprod));

  bpr_F0=sum(elem_prod(N_bpr_F0,wgt_mt));



FUNCTION get_selectivity


  //BLOCK 1 for selex.

  for (iyear=styr; iyear<=endyr_selex_phase1; iyear++)

  {

     sel_cH(iyear)=logistic(agebins, selpar_L50_cH1, selpar_slope_cH1);


     sel_mcvt(iyear)=logistic(agebins, selpar_L50_mcvt, selpar_slope_mcvt);

     //sel_mcvt(iyear)=logistic_exponential(agebins, selpar_L50_mcvt, selpar_slope_mcvt, selpar_sigma_mcvt,
selpar_afull_mcvt);
```

```
    //sel_vid(iyear)=logistic_exponential(agebins, selpar_L50_vid, selpar_slope_vid, selpar_sigma_vid,
selpar_afull_vid);

    //sel_vid(iyear)=logistic(agebins, selpar_L50_vid, selpar_slope_vid);

    //sel_vid(iyear)=sel_mcvt(iyear);


    sel_HB(iyear)=logistic_double(agebins, selpar_L51_HB1, selpar_slope1_HB1, selpar_L52_HB1,
selpar_slope2_HB1);

   //sel_HB(iyear)=logistic(agebins, selpar_L50_HB1, selpar_slope_HB1);

   //sel_HB(iyear)=logistic_exponential(agebins, selpar_L50_HB1, selpar_slope_HB1, selpar_sigma_HB1,
selpar_afull_HB1);


    sel_HB_D(iyear)=logistic_exponential(agebins, selpar_L50_HB_D, selpar_slope_HB_D, selpar_sigma_HB_D,
selpar_afull_HB_D);


    sel_GR(iyear)=logistic_double(agebins, selpar_L51_GR, selpar_slope1_GR, selpar_L52_GR,
selpar_slope2_GR);

   //sel_GR(iyear)=logistic(agebins, selpar_L50_GR, selpar_slope_GR);


    sel_GR_D(iyear)=sel_HB_D(iyear);

  }


//year-specific age at peak selectivity based on inspection of HB age comps (HB1=peak at age 3, HB2=peak at age 6

//if no annual age comp available then assume peak at age 3 (HB1); above loop


//     sel_HB(1991)=logistic_double(agebins, selpar_L50_HB2, selpar_slope_HB2, selpar_afull_HB2,
selpar_sigma_HB2);

//     sel_HB(2007)=logistic_double(agebins, selpar_L50_HB2, selpar_slope_HB2, selpar_afull_HB2,
selpar_sigma_HB2);

//     sel_HB(2008)=logistic_double(agebins, selpar_L50_HB2, selpar_slope_HB2, selpar_afull_HB2,
selpar_sigma_HB2);

//     sel_HB(2009)=logistic_double(agebins, selpar_L50_HB2, selpar_slope_HB2, selpar_afull_HB2,
selpar_sigma_HB2);

//     sel_HB(2011)=logistic_double(agebins, selpar_L50_HB2, selpar_slope_HB2, selpar_afull_HB2,
selpar_sigma_HB2);
```

```
//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_mcvt(iyear)=logistic_exponential(agebins, selpar_L50_mcvt, selpar_slope_mcvt, selpar_sigma_mcvt,
selpar_afull_mcvt);
//     sel_HB(iyear)=logistic(agebins, selpar_L50_HB2, selpar_slope_HB2);

//     sel_mcvt(iyear)(8,nages)=sel_mcvt(iyear)(7);

//   }

//  //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_mcvt(iyear)=logistic_exponential(agebins, selpar_L50_mcvt, selpar_slope_mcvt, selpar_sigma_mcvt,
selpar_afull_mcvt);
//     sel_HB(iyear)=logistic(agebins, selpar_L50_HB3, selpar_slope_HB3);

//     sel_mcvt(iyear)(8,nages)=sel_mcvt(iyear)(7);
//    }

//  sel_initial=sel_rec(styr);

FUNCTION get_mortality

  Fsum.initialize();
```

```
 Fapex.initialize();
 F.initialize();
 //initialization F is avg from first 3 yrs of observed landings
 log_F_dev_init_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)); //}
  // if (iyear<styr_cH_L){F_cH_out(iyear)=mfexp(log_avg_F_cH+log_F_dev_init_cH);}
     F_cH(iyear)=sel_cH(iyear)*F_cH_out(iyear);
     Fsum(iyear)+=F_cH_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)); //}
  // if (iyear<styr_HB_L){F_HB_out(iyear)=mfexp(log_avg_F_HB+log_F_dev_init_HB);}
     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)); //}
  // if (iyear<styr_GR_L){F_GR_out(iyear)=mfexp(log_avg_F_GR+log_F_dev_init_GR);}
     F_GR(iyear)=sel_GR(iyear)*F_GR_out(iyear); //general rec shares headboat selex
     Fsum(iyear)+=F_GR_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)); //}
  // if (iyear<styr_HB_D){F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_dev_init_HB_D);}
    F_HB_D(iyear)=sel_HB_D(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)); //}
  // if (iyear<styr_GR_D){F_GR_D_out(iyear)=mfexp(log_avg_F_GR_D+log_F_dev_init_GR_D);}
    F_GR_D(iyear)=sel_HB_D(iyear)*F_GR_D_out(iyear); //general rec discards shares headboat discards 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_HB(iyear);
  F(iyear)+=F_GR(iyear);
  F(iyear)+=F_HB_D(iyear);
  F(iyear)+=F_GR_D(iyear);

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

 }  //end iyear


FUNCTION get_bias_corr
```

```
 var_rec_dev=norm2(log_rec_dev(styr_rec_dev,endyr_rec_dev)-
          sum(log_rec_dev(styr_rec_dev,endyr_rec_dev))/nyrs_rec)
          /(nyrs_rec-1.0);
 //if (set_BiasCor <= 0.0) {BiasCor=mfexp(var_rec_dev/2.0);}   //bias correction based on empirical residuals
 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=(R0/((5.0*steep-1.0)*spr_F0))*
 //             (BiasCor*4.0*steep*spr_F0-spr_F0*(1.0-steep));
 //R_virgin=R0/(spr_F0/spr_F0)*BiasCor*(1.0+log(spr_F0/spr_F0)/steep); //Ricker

 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)+
          mfexp(log_avg_F_HB_D+log_F_dev_init_HB_D)+
          mfexp(log_avg_F_GR_D+log_F_dev_init_GR_D);
```

```
 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_init_HB_D_prop=mfexp(log_avg_F_HB_D+log_F_dev_init_HB_D)/F_init_denom;
 F_init_GR_D_prop=mfexp(log_avg_F_GR_D+log_F_dev_init_GR_D)/F_init_denom;

 F_initial=sel_cH(styr)*F_init*F_init_cH_prop+
         sel_HB(styr)*F_init*F_init_HB_prop+
         sel_GR(styr)*F_init*F_init_GR_prop+
         sel_HB_D(styr)*F_init*F_init_HB_prop+
         sel_GR_D(styr)*F_init*F_init_GR_prop;

 Z_initial=M+F_initial;

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

 //if (styr==styr_rec_dev) {R1=(R0/((5.0*steep-1.0)*spr_initial))*
 //            (4.0*steep*spr_initial-spr_F0*(1.0-steep));} //without bias correction (deviation added later)
 //else {R1=(R0/((5.0*steep-1.0)*spr_initial))*
 //            (BiasCor*4.0*steep*spr_initial-spr_F0*(1.0-steep));} //with bias correction
```

```
  if (styr==styr_rec_dev) {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, 1.0, SR_switch);} //without bias
correction (deviation added later)

  else {R1=SR_eq_func(R0, steep, spr_F0, spr_initial, BiasCor, SR_switch);} //with bias correction

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


//Compute equilibrium age structure for first year

  N_initial_eq(1)=R1;

  for (iage=2; iage<=nages; iage++)

  {

    N_initial_eq(iage)=N_initial_eq(iage-1)*

        mfexp(-1.0*(Z_initial(iage-1)));

  }

  //plus group calculation

  N_initial_eq(nages)=N_initial_eq(nages)/(1.0-mfexp(-1.0*Z_initial(nages))); //plus group


//Add deviations to initial equilibrium N

  N(styr)(2,nages)=elem_prod(N_initial_eq(2,nages),mfexp(log_Nage_dev));


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

  else {N(styr,1)=N_initial_eq(1);}


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

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


  SSB(styr)=sum(elem_prod(N_spawn(styr),reprod));  //KC4

  MatFemB(styr)=sum(elem_prod(N_spawn(styr),reprod2));

  B_q_DD(styr)=sum(elem_prod(N(styr)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
```

```
//Rest of years

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

  {

    if(iyear<(styr_rec_dev-1)||iyear>(endyr_rec_dev-1)) //recruitment follows S-R curve (with bias correction) exactly

    {

       N(iyear+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch);

       N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));

       N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages)); //plus group

       N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year

       N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time

       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod)); //KC

//KC       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod(iyear+1)));

//KC       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2(iyear+1)));

       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));  //KC

       B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));

    }

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

    {

       N(iyear+1,1)=SR_func(R0, steep, spr_F0, SSB(iyear),SR_switch)*mfexp(log_rec_dev(iyear+1));

       N(iyear+1)(2,nages)=++elem_prod(N(iyear)(1,nages-1),(mfexp(-1.*Z(iyear)(1,nages-1))));

       N(iyear+1,nages)+=N(iyear,nages)*mfexp(-1.*Z(iyear,nages)); //plus group

       N_mdyr(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*0.5))); //mid year

       N_spawn(iyear+1)(1,nages)=elem_prod(N(iyear+1)(1,nages),(mfexp(-1.*(Z(iyear+1)(1,nages))*spawn_time_frac))); //peak spawning time

       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod));  //KC4

//KC       SSB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod(iyear+1)));

//KC       MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2(iyear+1)));
```

```
      MatFemB(iyear+1)=sum(elem_prod(N_spawn(iyear+1),reprod2));
      B_q_DD(iyear+1)=sum(elem_prod(N(iyear+1)(set_q_DD_stage,nages),wgt_mt(set_q_DD_stage,nages)));
    }
  }

  //last year (projection) has no recruitment variability
  N(endyr+1,1)=BiasCor*SR_func(R0, steep, spr_F0, SSB(endyr),SR_switch);
  N(endyr+1)(2,nages)=++elem_prod(N(endyr)(1,nages-1),(mfexp(-1.*Z(endyr)(1,nages-1))));
  N(endyr+1,nages)+=N(endyr,nages)*mfexp(-1.*Z(endyr,nages)); //plus group


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_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_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++)
 {
  L_cH_klb(iyear)=elem_prod(L_cH_num(iyear),wholewgt_cH_klb(iyear));     //in 1000 lb whole weight
  L_HB_klb(iyear)=elem_prod(L_HB_num(iyear),wholewgt_HB_klb(iyear));     //in 1000 lb whole weight
  L_GR_klb(iyear)=elem_prod(L_GR_num(iyear),wholewgt_GR_klb(iyear));     //in 1000 lb whole weight

  pred_cH_L_klb(iyear)=sum(L_cH_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_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),wholewgt_HB_D_klb(iyear))); //annual dead discards in 1000 lb whole (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),wholewgt_GR_D_klb(iyear))); //annual dead discards in 1000 lb whole (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+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_mcvt_cpue; iyear<=endyr_mcvt_cpue; iyear++) //KC added this section
    {   if (iyear>styr_mcvt_cpue & iyear <=2003)
      {//q_rate_fcn_mcvt(iyear)=(1.0+q_rate)*q_rate_fcn_mcvt(iyear-1); //compound
        q_rate_fcn_mcvt(iyear)=(1.0+(iyear-styr_mcvt_cpue)*q_rate)*q_rate_fcn_mcvt(styr_mcvt_cpue);  //linear
      }
      if (iyear>2003) {q_rate_fcn_mcvt(iyear)=q_rate_fcn_mcvt(iyear-1);}
    }
//    for (iyear=styr_vid_cpue; iyear<=endyr_vid_cpue; iyear++)
//    {   if (iyear>styr_vid_cpue & iyear <=2003)
//      {//q_rate_fcn_vid(iyear)=(1.0+q_rate)*q_rate_fcn_vid(iyear-1); //compound
//        q_rate_fcn_vid(iyear)=(1.0+(iyear-styr_vid_cpue)*q_rate)*q_rate_fcn_vid(styr_vid_cpue);  //linear
//      }
//      if (iyear>2003) {q_rate_fcn_vid(iyear)=q_rate_fcn_vid(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

 //Get density dependence scalar (=1.0 if density independent model is used)
  if (q_DD_beta>0.0)
  {
    B_q_DD+=dzero;
    for (iyear=styr;iyear<=endyr;iyear++)
       {q_DD_fcn(iyear)=pow(B0_q_DD,q_DD_beta)*pow(B_q_DD(iyear),-q_DD_beta);}
        //{q_DD_fcn(iyear)=1.0+4.0/(1.0+mfexp(0.75*(B_q_DD(iyear)-0.1*B0_q_DD))); }
  }



FUNCTION get_indices

//---Predicted CPUEs------------------------

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

```
 //mcvt  cpue
  q_mcvt(styr_mcvt_cpue)=mfexp(log_q_mcvt);
  for (iyear=styr_mcvt_cpue; iyear<=endyr_mcvt_cpue; iyear++)
  {//index in weight units. original index in lb and re-scaled. predicted in klb whole weight, but difference in lb and klb is absorbed by q
     //N_mcvt(iyear)=elem_prod(elem_prod(N_mdyr(iyear),sel_mcvt(iyear)),wholewgt_mcvt_klb(iyear));
     N_mcvt(iyear)=elem_prod(N_mdyr(iyear),sel_mcvt(iyear));
     pred_mcvt_cpue(iyear)=q_mcvt(iyear)*q_rate_fcn_mcvt(iyear)*q_DD_fcn(iyear)*sum(N_mcvt(iyear));
     if (iyear<endyr_mcvt_cpue){q_mcvt(iyear+1)=q_mcvt(iyear)*mfexp(q_RW_log_dev_mcvt(iyear));}
  }

//video  cpue
//  q_vid(styr_vid_cpue)=mfexp(log_q_vid);
//  for (iyear=styr_vid_cpue; iyear<=endyr_vid_cpue; iyear++)
//  {//index in weight units. original index in lb and re-scaled. predicted in klb whole weight, but difference in lb and klb is absorbed by q
//     N_vid(iyear)=elem_prod(N_mdyr(iyear),sel_vid(iyear));
//     //N_vid(iyear)=elem_prod(N_mdyr(iyear),sel_vid(iyear));
//     pred_vid_cpue(iyear)=q_vid(iyear)*q_rate_fcn_vid(iyear)*q_DD_fcn(iyear)*sum(N_vid(iyear));
//     if (iyear<endyr_vid_cpue){q_vid(iyear+1)=q_vid(iyear)*mfexp(q_RW_log_dev_vid(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

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

 //SERFS chevron trap
  for (iyear=1;iyear<=nyr_mcvt_lenc;iyear++)
  {pred_mcvt_lenc(iyear)=(N_mcvt(yrs_mcvt_lenc(iyear))*lenprob_mcvt)/sum(N_mcvt(yrs_mcvt_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)));}

 //headboat discards
  for (iyear=1;iyear<=nyr_HB_D_lenc;iyear++)

{pred_HB_D_lenc(iyear)=(D_HB_num(yrs_HB_D_lenc(iyear))*lenprob_HB_D)/sum(D_HB_num(yrs_HB_D_len
c(iyear)));}
```

```
 //GR (MRIP)
  for (iyear=1;iyear<=nyr_GR_lenc;iyear++)
  {pred_GR_lenc(iyear)=(L_GR_num(yrs_GR_lenc(iyear))*lenprob_GR)/sum(L_GR_num(yrs_GR_lenc(iyear)));}

      // cout << pred_HB_D_lenc <<endl;


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)));
   //ErrorFree_cH_agec(iyear)=elem_prod(N(yrs_cH_agec(iyear)),sel_cH(yrs_cH_agec(iyear)));
   pred_cH_agec_allages(iyear)=age_error*(ErrorFree_cH_agec(iyear)/sum(ErrorFree_cH_agec(iyear)));
   for (iage=1; iage<=nages_agec; iage++) {pred_cH_agec(iyear,iage)=pred_cH_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++)
{pred_cH_agec(iyear,nages_agec)+=pred_cH_agec_allages(iyear,iage);} //plus group
  }

  //SERFS chevron trap
  for (iyear=1;iyear<=nyr_mcvt_agec;iyear++)
  {
   ErrorFree_mcvt_agec(iyear)=N_mcvt(yrs_mcvt_agec(iyear))/sum(N_mcvt(yrs_mcvt_agec(iyear)));
   //ErrorFree_mcvt_agec(iyear)=elem_prod(N(yrs_mcvt_agec(iyear)),sel_mcvt(yrs_mcvt_agec(iyear)));
   pred_mcvt_agec_allages(iyear)=age_error*(ErrorFree_mcvt_agec(iyear)/sum(ErrorFree_mcvt_agec(iyear)));
   for (iage=1; iage<=nages_agec; iage++) {pred_mcvt_agec(iyear,iage)=pred_mcvt_agec_allages(iyear,iage);}
   for (iage=(nages_agec+1); iage<=nages; iage++)
{pred_mcvt_agec(iyear,nages_agec)+=pred_mcvt_agec_allages(iyear,iage);} //plus group
```

```
  }

 //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);} //plus group
  }

 // //GR (MRIP)
 // for (iyear=1;iyear<=nyr_GR_agec;iyear++)
  // {
   // ErrorFree_GR_agec(iyear)=L_GR_num(yrs_GR_agec(iyear))/sum(L_GR_num(yrs_GR_agec(iyear)));
   // pred_GR_agec_allages(iyear)=age_error*ErrorFree_GR_agec(iyear);
   // for (iage=1; iage<=nages_agec; iage++) {pred_GR_agec(iyear,iage)=pred_GR_agec_allages(iyear,iage);}
   // for (iage=(nages_agec+1); iage<=nages; iage++)
{pred_GR_agec(iyear,nages_agec)+=pred_GR_agec_allages(iyear,iage);} //plus group
  // }


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

FUNCTION get_weighted_current
  F_temp_sum=0.0;
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_cH+
       sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
  F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_HB+
```

```
      sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
 F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_GR+
      sum(log_F_dev_GR((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
 F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_HB_D+
      sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);
 F_temp_sum+=mfexp((selpar_n_yrs_wgted*log_avg_F_GR_D+
      sum(log_F_dev_GR_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted);

 F_cH_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cH+
      sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
 F_HB_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_HB+
      sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
 F_GR_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_GR+
      sum(log_F_dev_GR((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
 F_HB_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_HB_D+
      sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;
 F_GR_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_GR_D+
      sum(log_F_dev_GR_D((endyr-selpar_n_yrs_wgted+1),endyr)))/selpar_n_yrs_wgted)/F_temp_sum;

 log_F_dev_end_cH=sum(log_F_dev_cH((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
 log_F_dev_end_HB=sum(log_F_dev_HB((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
 log_F_dev_end_GR=sum(log_F_dev_GR((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

 log_F_dev_end_HB_D=sum(log_F_dev_HB_D((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;
 log_F_dev_end_GR_D=sum(log_F_dev_GR_D((endyr-selpar_n_yrs_wgted+1),endyr))/selpar_n_yrs_wgted;

 F_end_L=sel_cH(endyr)*mfexp(log_avg_F_cH+log_F_dev_end_cH)+
       sel_HB(endyr)*mfexp(log_avg_F_HB+log_F_dev_end_HB)+
       sel_GR(endyr)*mfexp(log_avg_F_GR+log_F_dev_end_GR);
```

```
 F_end_D=sel_HB_D(endyr)*mfexp(log_avg_F_HB_D+log_F_dev_end_HB_D)+
       sel_GR_D(endyr)*mfexp(log_avg_F_GR_D+log_F_dev_end_GR_D);

 F_end=F_end_L+F_end_D;
 F_end_apex=max(F_end);

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

 wgt_wgted_L_denom=F_cH_prop+F_HB_prop+F_GR_prop;
 wgt_wgted_L_klb=F_cH_prop/wgt_wgted_L_denom*wholewgt_cH_klb(endyr)+
            F_HB_prop/wgt_wgted_L_denom*wholewgt_HB_klb(endyr)+
            F_GR_prop/wgt_wgted_L_denom*wholewgt_GR_klb(endyr);

 wgt_wgted_D_denom=F_HB_D_prop+F_GR_D_prop;
 wgt_wgted_D_klb=F_HB_D_prop/wgt_wgted_D_denom*wholewgt_HB_D_klb(endyr)+
           F_GR_D_prop/wgt_wgted_D_denom*wholewgt_GR_D_klb(endyr);


FUNCTION get_msy

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

```
  F_D_age_msy=0.0;

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

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

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

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

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

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

```
                (1.-mfexp(-1.0*Z_age_msy(iage)));
  }

  SSB_eq(ff)=sum(elem_prod(N_age_msy_spawn,reprod));
  B_eq(ff)=sum(elem_prod(N_age_msy,wgt_mt));
  L_eq_klb(ff)=sum(elem_prod(L_age_msy,wgt_wgted_L_klb)); //in whole weight
  L_eq_knum(ff)=sum(L_age_msy)/1000.0;
  D_eq_klb(ff)=sum(elem_prod(D_age_msy,wgt_wgted_D_klb)); //in whole weight
  D_eq_knum(ff)=sum(D_age_msy)/1000.0;
}

msy_klb_out=max(L_eq_klb); //msy in whole weight

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

```
//-----------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------

FUNCTION get_per_recruit_stuff


 //static per-recruit stuff


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



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


  F_L_age_spr=F_spr(ff)*sel_wgted_L;
```

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

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

  spr_spr(ff)=sum(elem_prod(N_age_spr_spawn,reprod));  //KC4
  L_spr(ff)=0.0;
  for (iage=1; iage<=nages; iage++)
  {
    L_age_spr(iage)=N_age_spr(iage)*(F_L_age_spr(iage)/Z_age_spr(iage))*
                  (1.-mfexp(-1.*Z_age_spr(iage)));
    L_spr(ff)+=L_age_spr(iage)*wgt_wgted_L_klb(iage)*1000.0; //in lb gutted wgt
  }
 }
 spr_ratio=spr_spr/spr_F0;              //added 1-19-16 here to...
 F20_dum=min(fabs(spr_ratio-0.2));
 F30_dum=min(fabs(spr_ratio-0.3));
 F40_dum=min(fabs(spr_ratio-0.4));
 for(ff=1; ff<=n_iter_spr; ff++)
 {
    if (fabs(spr_ratio(ff)-0.2)==F20_dum) {F20_out=F_spr(ff);
```

```
            }

            if (fabs(spr_ratio(ff)-0.3)==F30_dum) {
                    F30_out=F_spr(ff);
      SSB_F30_out=SSB_eq(ff);  //NOTE, this works bc F grid for msy calcs is the same as for spr calcs
                    B_F30_out=B_eq(ff);
      R_F30_out=R_eq(ff);
      L_F30_knum_out=L_eq_knum(ff);
      L_F30_klb_out=L_eq_klb(ff);
                    D_F30_knum_out=D_eq_knum(ff);
      D_F30_klb_out=D_eq_klb(ff);
            }

            if (fabs(spr_ratio(ff)-0.4)==F40_dum) {
                    F40_out=F_spr(ff);
            }
  }  //here.

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

FUNCTION get_miscellaneous_stuff

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

//  len_cv=elem_div(len_sd,meanlen_FL);
```

```
 len_sd=elem_prod(len_cv,meanlen_FL);


 for(iage=1;iage<=nages;iage++)                //Added this loop to allow plotting of both Population and Landings length at age
            {
//        len_sd(iage)=len_sd_val;
//        len_cv(iage)=len_sd(iage)/meanlen_FL(iage);
//        len_sd_L(iage)=len_sd_val_L;
//        len_cv_L(iage)=len_sd_L(iage)/meanlen_FL_L(iage);
//        len_sd_mcvt(iage)=len_sd_val_mcvt;
//        len_cv_mcvt(iage)=len_sd_mcvt(iage)/meanlen_FL_mcvt(iage);


            len_cv(iage)=len_cv_val;
            len_sd(iage)=meanlen_FL(iage)*len_cv(iage);
            len_cv_L(iage)=len_cv_val_L;
            len_sd_L(iage)=meanlen_FL_L(iage)*len_cv_L(iage);
        len_cv_mcvt(iage)=len_cv_val_mcvt;
        len_sd_mcvt(iage)=meanlen_FL_mcvt(iage)*len_cv_mcvt(iage);
            }


 //compute total landings- and discards-at-age in 1000 fish and klb whole weight
 L_total_num.initialize();
 L_total_klb.initialize();
 L_total_knum_yr.initialize();
 L_total_klb_yr.initialize();
 D_total_num.initialize();
 D_total_klb.initialize();
 D_total_knum_yr.initialize();
 D_total_klb_yr.initialize();
```

```
for(iyear=styr; iyear<=endyr; iyear++)
{
        L_total_klb_yr(iyear)=pred_cH_L_klb(iyear)+pred_HB_L_klb(iyear)+pred_GR_L_klb(iyear);
        L_total_knum_yr(iyear)=pred_cH_L_knum(iyear)+pred_HB_L_knum(iyear)+pred_GR_L_knum(iyear);

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

    if (iyear>=styr_HB_D && iyear<=endyr_HB_D)
    {
     D_total_knum_yr(iyear)+=pred_HB_D_knum(iyear);
     D_total_klb_yr(iyear)+=pred_HB_D_klb(iyear);
     D_HB_klb(iyear)=elem_prod(D_HB_num(iyear),wholewgt_HB_D_klb(iyear));
    }

    if (iyear>=styr_GR_D && iyear<=endyr_GR_D)
    {
     D_total_knum_yr(iyear)+=pred_GR_D_knum(iyear);
     D_total_klb_yr(iyear)+=pred_GR_D_klb(iyear);
     D_GR_klb(iyear)=elem_prod(D_GR_num(iyear),wholewgt_GR_D_klb(iyear));
    }
   }

L_total_num=L_cH_num+L_HB_num+L_GR_num;   //added landings at age in number fish
L_total_klb=L_cH_klb+L_HB_klb+L_GR_klb;   //landings at age in klb whole weight

D_total_num=(D_HB_num+D_GR_num);          //discards at age in number fish
```

```
 D_total_klb=D_HB_klb+D_GR_klb;          //discards at age in klb whole weight



 //Time series of interest

 B(endyr+1)=elem_prod(N(endyr+1),wgt_mt);
 totN(endyr+1)=sum(N(endyr+1));
 totB(endyr+1)=sum(B(endyr+1));
 //N_spawn(endyr+1)=N(endyr+1);
 //SSB(endyr+1)=sum(elem_prod(N_spawn(endyr+1),reprod));
 //MatFemB(endyr+1)=sum(elem_prod(N_spawn(endyr+1),reprod2));
 rec=column(N,1);
 SdS0=SSB/S0;

//  steep_sd=steep;
//  fullF_sd=Fsum;
 Fend_mean_temp=1.0;   //added 1-19-16 here to...
 for (iyear=1; iyear<=selpar_n_yrs_wgted; iyear++) {Fend_mean_temp*=Fapex(endyr-iyear+1);}
 Fend_mean=pow(Fend_mean_temp,(1.0/selpar_n_yrs_wgted));          //here.

 if(F_msy_out>0)
   {
    FdF_msy=Fapex/F_msy_out;
    FdF_msy_end=FdF_msy(endyr);
           FdF_msy_end_mean=Fend_mean/F_msy_out;  //added 1-19-16
    //FdF_msy_end_mean=pow((FdF_msy(endyr)*FdF_msy(endyr-1)*FdF_msy(endyr-2)),(1.0/3.0));
   }
 if(SSB_msy_out>0)
   {
```

```
    SdSSB_msy=SSB/SSB_msy_out;
    SdSSB_msy_end=SdSSB_msy(endyr);
   }
  if(F30_out>0)                //added 1-19-16  here to...
   {
            FdF30=Fapex/F30_out;
            FdF30_end_mean=Fend_mean/F30_out;
           }
  if(SSB_F30_out>0)
   {
    SdSSB_F30=SSB/SSB_F30_out;
            Sdmsst_F30=SSB/(smsy2msst*SSB_F30_out);  //KC-changed from smsy2msst75
    SdSSB_F30_end=SdSSB_F30(endyr);
            Sdmsst_F30_end=Sdmsst_F30(endyr);
   }                                                              //here
  //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_effective_sample_sizes
    neff_cH_lenc_allyr_out=missing;
    neff_mcvt_lenc_allyr_out=missing;
    neff_HB_lenc_allyr_out=missing;
    neff_HB_D_lenc_allyr_out=missing;
```

```
    neff_GR_lenc_allyr_out=missing;

    neff_cH_agec_allyr_out=missing;
    neff_mcvt_agec_allyr_out=missing;
    neff_HB_agec_allyr_out=missing;
    //neff_GR_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_mcvt_lenc; iyear++)
      {if (nsamp_mcvt_lenc(iyear)>=minSS_mcvt_lenc)
{neff_mcvt_lenc_allyr_out(yrs_mcvt_lenc(iyear))=multinom_eff_N(pred_mcvt_lenc(iyear),obs_mcvt_lenc(iyear));
}
       else {neff_mcvt_lenc_allyr_out(yrs_mcvt_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_HB_D_lenc; iyear++)
      {if (nsamp_HB_D_lenc(iyear)>=minSS_HB_D_lenc)
```

```
{neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=multinom_eff_N(pred_HB_D_lenc(iyear),obs_HB_D_lenc(iy
ear));}
        else {neff_HB_D_lenc_allyr_out(yrs_HB_D_lenc(iyear))=-99;}
        }

    for (iyear=1; iyear<=nyr_GR_lenc; iyear++)
      {if (nsamp_GR_lenc(iyear)>=minSS_GR_lenc)

{neff_GR_lenc_allyr_out(yrs_GR_lenc(iyear))=multinom_eff_N(pred_GR_lenc(iyear),obs_GR_lenc(iyear));}
        else {neff_GR_lenc_allyr_out(yrs_GR_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_mcvt_lenc; iyear++)
      {if (nsamp_mcvt_lenc(iyear)>=minSS_mcvt_lenc)

{neff_mcvt_lenc_allyr_out(yrs_mcvt_lenc(iyear))=multinom_eff_N(pred_mcvt_lenc(iyear),obs_mcvt_lenc(iyear));
}
        else {neff_mcvt_lenc_allyr_out(yrs_mcvt_lenc(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;}
        }

     // for (iyear=1; iyear<=nyr_GR_agec; iyear++)
       // {if (nsamp_GR_agec(iyear)>=minSS_GR_agec)
         //
{neff_GR_agec_allyr_out(yrs_GR_agec(iyear))=multinom_eff_N(pred_GR_agec(iyear),obs_GR_agec(iyear));}
        // else {neff_GR_agec_allyr_out(yrs_GR_agec(iyear))=-99;}
       // }

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

FUNCTION evaluate_objective_function
  //fval=square(xdum-9.0);

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

//---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_mcvt_cpue=0.0;
  f_mcvt_cpue=lk_lognormal(pred_mcvt_cpue, obs_mcvt_cpue, mcvt_cpue_cv, w_I_mcvt);
```

```
  fval+=f_mcvt_cpue;
  fval_data+=f_mcvt_cpue;

//  f_vid_cpue=0.0;
//  f_vid_cpue=lk_lognormal(pred_vid_cpue, obs_vid_cpue, vid_cpue_cv, w_I_vid);
//  fval+=f_vid_cpue;
//  fval_data+=f_vid_cpue;

//  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_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;

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

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

  //f_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_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
  f_cH_lenc=lk_robust_multinomial(nsamp_cH_lenc, pred_cH_lenc, obs_cH_lenc, nyr_cH_lenc, double(nlenbins),
minSS_cH_lenc, w_lc_cH);
  //f_cH_lenc=lk_multinomial(nsamp_cH_lenc, pred_cH_lenc, obs_cH_lenc, nyr_cH_lenc, minSS_cH_lenc,
w_lc_cH);
```

```
  fval+=f_cH_lenc;

  fval_data+=f_cH_lenc;


//  f_mcvt_lenc

  f_mcvt_lenc=lk_robust_multinomial(nsamp_mcvt_lenc, pred_mcvt_lenc, obs_mcvt_lenc, nyr_mcvt_lenc,
double(nlenbins), minSS_mcvt_lenc, w_lc_mcvt);

  //f_mcvt_lenc=lk_multinomial(nsamp_mcvt_lenc, pred_mcvt_lenc, obs_mcvt_lenc, nyr_mcvt_lenc,
minSS_mcvt_lenc, w_lc_mcvt);

  fval+=f_mcvt_lenc;

  fval_data+=f_mcvt_lenc;


//  f_HB_lenc

  f_HB_lenc=lk_robust_multinomial(nsamp_HB_lenc, pred_HB_lenc, obs_HB_lenc, nyr_HB_lenc,
double(nlenbins), minSS_HB_lenc, w_lc_HB);

  //f_HB_lenc=lk_multinomial(nsamp_HB_lenc, pred_HB_lenc, obs_HB_lenc, nyr_HB_lenc, minSS_HB_lenc,
w_lc_HB);

  fval+=f_HB_lenc;

  fval_data+=f_HB_lenc;


//  f_HB_lenc_D

  f_HB_D_lenc=lk_robust_multinomial(nsamp_HB_D_lenc, pred_HB_D_lenc, obs_HB_D_lenc, nyr_HB_D_lenc,
double(nlenbins), minSS_HB_D_lenc, w_lc_HB_D);

  //f_HB_D_lenc=lk_multinomial(nsamp_HB_D_lenc, pred_HB_D_lenc, obs_HB_D_lenc, nyr_HB_D_lenc,
minSS_HB_D_lenc, w_lc_HB_d);

  fval+=f_HB_D_lenc;

  fval_data+=f_HB_D_lenc;


//  f_GR_lenc

  f_GR_lenc=lk_robust_multinomial(nsamp_GR_lenc, pred_GR_lenc, obs_GR_lenc, nyr_GR_lenc,
double(nlenbins), minSS_GR_lenc, w_lc_GR);

  //f_GR_lenc=lk_multinomial(nsamp_GR_lenc, pred_GR_lenc, obs_GR_lenc, nyr_GR_lenc, minSS_GR_lenc,
w_lc_GR);

  fval+=f_GR_lenc;
```

```
  fval_data+=f_GR_lenc;

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


//  f_cH_agec

  f_cH_agec=lk_robust_multinomial(nsamp_cH_agec, pred_cH_agec, obs_cH_agec, nyr_cH_agec,
double(nages_agec), minSS_cH_agec, w_ac_cH);

  //f_cH_agec=lk_multinomial(nsamp_cH_agec, pred_cH_agec, obs_cH_agec, nyr_cH_agec, minSS_cH_agec,
w_ac_cH);

  fval+=f_cH_agec;

  fval_data+=f_cH_agec;


//  f_mcvt_agec

  f_mcvt_agec=lk_robust_multinomial(nsamp_mcvt_agec, pred_mcvt_agec, obs_mcvt_agec, nyr_mcvt_agec,
double(nages_agec), minSS_mcvt_agec, w_ac_mcvt);

  //f_mcvt_agec=lk_multinomial(nsamp_mcvt_agec, pred_mcvt_agec, obs_mcvt_agec, nyr_mcvt_agec,
minSS_mcvt_agec, w_ac_mcvt);

  fval+=f_mcvt_agec;

  fval_data+=f_mcvt_agec;


//  f_HB_agec

  f_HB_agec=lk_robust_multinomial(nsamp_HB_agec, pred_HB_agec, obs_HB_agec, nyr_HB_agec,
double(nages_agec), minSS_HB_agec, w_ac_HB);

  //f_HB_agec=lk_multinomial(nsamp_HB_agec, pred_HB_agec, obs_HB_agec, nyr_HB_agec, minSS_HB_agec,
w_ac_HB);

  fval+=f_HB_agec;

  fval_data+=f_HB_agec;


// //  f_GR_agec

  // f_GR_agec=lk_robust_multinomial(nsamp_GR_agec, pred_GR_agec, obs_GR_agec, nyr_GR_agec,
double(nages_agec), minSS_GR_agec, w_ac_GR);

  // //f_GR_agec=lk_multinomial(nsamp_GR_agec, pred_GR_agec, obs_GR_agec, nyr_GR_agec, minSS_GR_agec,
w_ac_GR);

  // fval+=f_GR_agec;
```

```
  // fval_data+=f_GR_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_sigma_sq=square(rec_sigma);
  rec_logL_add=nyrs_rec*log(rec_sigma);
  f_rec_dev=(square(log_rec_dev(styr_rec_dev) + rec_sigma_sq/2.0)/(2.0*rec_sigma_sq));
  for(iyear=(styr_rec_dev+1); iyear<=endyr_rec_dev; iyear++)
  {f_rec_dev+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
              (2.0*rec_sigma_sq));}
  f_rec_dev+=rec_logL_add;
  fval+=w_rec*f_rec_dev;


  f_rec_dev_early=0.0; //possible extra constraint on early rec deviations
  if (w_rec_early>0.0)
    { if (styr_rec_dev<endyr_rec_phase1)
       {
        for(iyear=styr_rec_dev; iyear<=endyr_rec_phase1; iyear++)
        //{f_rec_dev_early+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
        //                (2.0*rec_sigma_sq)) + rec_logL_add;}
        {f_rec_dev_early+=square(log_rec_dev(iyear));}
       }
  fval+=w_rec_early*f_rec_dev_early;
  }
```

```
 f_rec_dev_end=0.0; //possible extra constraint on ending rec deviations
 if (w_rec_end>0.0)
 { if (endyr_rec_phase2<endyr_rec_dev)
      {
       for(iyear=(endyr_rec_phase2+1); iyear<=endyr_rec_dev; iyear++)
       //{f_rec_dev_end+=(square(log_rec_dev(iyear)-R_autocorr*log_rec_dev(iyear-1) + rec_sigma_sq/2.0)/
       //              (2.0*rec_sigma_sq)) + rec_logL_add;}
       {f_rec_dev_end+=square(log_rec_dev(iyear));}
      }
    fval+=w_rec_end*f_rec_dev_end;
  }

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

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

//  //Random walk components of fishery dependent indices
```

```
//  f_HB_RW_cpue=0.0;
//  for (iyear=styr_HB_cpue; 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=0.0;
  f_priors+=neg_log_prior(Linf,set_Linf(5),set_Linf(6),set_Linf(7));
  f_priors+=neg_log_prior(K,set_K(5),set_K(6),set_K(7));
  f_priors+=neg_log_prior(t0,set_t0(5),set_t0(6),set_t0(7));
  f_priors+=neg_log_prior(len_cv_val,set_len_cv(5),set_len_cv(6),set_len_cv(7));

  f_priors+=neg_log_prior(M_constant,set_M_constant(5),set_M_constant(6),set_M_constant(7));

  f_priors+=neg_log_prior(Linf_L,set_Linf_L(5),set_Linf_L(6),set_Linf_L(7));
  f_priors+=neg_log_prior(K_L,set_K_L(5),set_K_L(6),set_K_L(7));
  f_priors+=neg_log_prior(t0_L,set_t0_L(5),set_t0_L(6),set_t0_L(7));
  f_priors+=neg_log_prior(len_cv_val_L,set_len_cv_L(5),set_len_cv_L(6),set_len_cv_L(7));

  f_priors+=neg_log_prior(Linf_mcvt,set_Linf_mcvt(5),set_Linf_mcvt(6),set_Linf_mcvt(7));
  f_priors+=neg_log_prior(K_mcvt,set_K_mcvt(5),set_K_mcvt(6),set_K_mcvt(7));
  f_priors+=neg_log_prior(t0_mcvt,set_t0_mcvt(5),set_t0_mcvt(6),set_t0_mcvt(7));
  f_priors+=neg_log_prior(len_cv_val_mcvt,set_len_cv_mcvt(5),set_len_cv_mcvt(6),set_len_cv_mcvt(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_mcvt,set_selpar_L50_mcvt(5), set_selpar_L50_mcvt(6),
set_selpar_L50_mcvt(7));

  f_priors+=neg_log_prior(selpar_slope_mcvt,set_selpar_slope_mcvt(5), set_selpar_slope_mcvt(6),
set_selpar_slope_mcvt(7));


  f_priors+=neg_log_prior(selpar_L51_HB1,set_selpar_L51_HB1(5), set_selpar_L51_HB1(6),
set_selpar_L51_HB1(7));

  f_priors+=neg_log_prior(selpar_slope1_HB1,set_selpar_slope1_HB1(5), set_selpar_slope1_HB1(6),
set_selpar_slope1_HB1(7));

  f_priors+=neg_log_prior(selpar_L52_HB1,set_selpar_L52_HB1(5), set_selpar_L52_HB1(6),
set_selpar_L52_HB1(7));

  f_priors+=neg_log_prior(selpar_slope2_HB1,set_selpar_slope2_HB1(5), set_selpar_slope2_HB1(6),
set_selpar_slope2_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_afull_HB2,set_selpar_afull_HB2(5), set_selpar_afull_HB2(6),
set_selpar_afull_HB2(7));

  f_priors+=neg_log_prior(selpar_sigma_HB2,set_selpar_sigma_HB2(5), set_selpar_sigma_HB2(6),
set_selpar_sigma_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(selpar_L50_HB_D,set_selpar_L50_HB_D(5), set_selpar_L50_HB_D(6),
set_selpar_L50_HB_D(7));

  f_priors+=neg_log_prior(selpar_slope_HB_D,set_selpar_slope_HB_D(5), set_selpar_slope_HB_D(6),
set_selpar_slope_HB_D(7));

  f_priors+=neg_log_prior(selpar_afull_HB_D,set_selpar_afull_HB_D(5), set_selpar_afull_HB_D(6),
set_selpar_afull_HB_D(7));

  f_priors+=neg_log_prior(selpar_sigma_HB_D,set_selpar_sigma_HB_D(5), set_selpar_sigma_HB_D(6),
set_selpar_sigma_HB_D(7));


  f_priors+=neg_log_prior(selpar_L51_GR,set_selpar_L51_GR(5), set_selpar_L51_GR(6), set_selpar_L51_GR(7));

  f_priors+=neg_log_prior(selpar_slope1_GR,set_selpar_slope1_GR(5), set_selpar_slope1_GR(6),
set_selpar_slope1_GR(7));

  f_priors+=neg_log_prior(selpar_L52_GR,set_selpar_L52_GR(5), set_selpar_L52_GR(6), set_selpar_L52_GR(7));

  f_priors+=neg_log_prior(selpar_slope2_GR,set_selpar_slope2_GR(5), set_selpar_slope2_GR(6),
set_selpar_slope2_GR(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_mcvt,set_log_q_mcvt(5),set_log_q_mcvt(6),set_log_q_mcvt(7));

//  f_priors+=neg_log_prior(log_q_vid,set_log_q_vid(5),set_log_q_vid(6),set_log_q_vid(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));
//  f_priors+=neg_log_prior(log_avg_F_cH,set_log_avg_F_cH(5),set_log_avg_F_cH(6),set_log_avg_F_cH(7));
//  f_priors+=neg_log_prior(log_avg_F_cL,set_log_avg_F_cL(5),set_log_avg_F_cL(6),set_log_avg_F_cL(7));
//  f_priors+=neg_log_prior(log_avg_F_HB,set_log_avg_F_HB(5),set_log_avg_F_HB(6),set_log_avg_F_HB(7));
//  f_priors+=neg_log_prior(log_avg_F_GR,set_log_avg_F_GR(5),set_log_avg_F_GR(6),set_log_avg_F_GR(7));

  fval+=f_priors;



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

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

//--------------------------------------------------------------------------------
//Logistic-exponential: 4 parameters (but 1 is fixed)
FUNCTION dvar_vector logistic_exponential(const dvar_vector& ages, const dvariable& L50, const dvariable&
slope, const dvariable& sigma, const dvariable& joint)
  //ages=vector of ages, L50=age at 50% sel (ascending limb), slope=rate of increase, sigma=controls rate of descent
(descending)
  //joint=age to join curves
  RETURN_ARRAYS_INCREMENT();
```

```
 dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());
 Sel_Tmp=1.0;
 for (iage=1; iage<=nages; iage++)
 {
  if (ages(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope*(ages(iage)-L50)));}
  if (ages(iage)>joint){Sel_Tmp(iage)=mfexp(-1.*square((ages(iage)-joint)/sigma));}
 }
 Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
 RETURN_ARRAYS_DECREMENT();
 return Sel_Tmp;


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

//Logistic function: 4 parameters

FUNCTION dvar_vector logistic_double(const dvar_vector& ages, const dvariable& L501, const dvariable&
slope1, const dvariable& L502, const dvariable& slope2)

  //ages=vector of ages, L50=age at 50% selectivity, slope=rate of increase, L502=age at 50% decrease additive to
L501, slope2=slope of decrease

 RETURN_ARRAYS_INCREMENT();

 dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());

 Sel_Tmp=elem_prod( (1./(1.+mfexp(-1.*slope1*(ages-L501)))),(1.-(1./(1.+mfexp(-1.*slope2*(ages-
(L501+L502)))))) );

 Sel_Tmp=Sel_Tmp/max(Sel_Tmp);

 RETURN_ARRAYS_DECREMENT();

 return Sel_Tmp;


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

//Jointed logistic function: 6 parameters (increasing and decreasing logistics joined at peak selectivity)

FUNCTION dvar_vector logistic_joint(const dvar_vector& ages, const dvariable& L501, const dvariable& slope1,
const dvariable& L502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)

  //ages=vector of ages, L501=age at 50% sel (ascending limb), slope1=rate of increase,L502=age at 50% sel
(descending), slope1=rate of increase (ascending),
```

```
  //satval=saturation value of descending limb, joint=location in age vector to join curves (may equal age or age + 1 if age-0 is included)

  RETURN_ARRAYS_INCREMENT();

  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());

  Sel_Tmp=1.0;

  for (iage=1; iage<=nages; iage++)

  {

   if (double(iage)<joint) {Sel_Tmp(iage)=1./(1.+mfexp(-1.*slope1*(ages(iage)-L501)));}

   if (double(iage)>joint){Sel_Tmp(iage)=1.0-(1.0-satval)/(1.+mfexp(-1.*slope2*(ages(iage)-L502)));}

  }

  Sel_Tmp=Sel_Tmp/max(Sel_Tmp);

  RETURN_ARRAYS_DECREMENT();

  return Sel_Tmp;


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

//Double Gaussian function: 6 parameters (as in SS3)

FUNCTION dvar_vector gaussian_double(const dvar_vector& ages, const dvariable& peak, const dvariable& top, const dvariable& ascwid, const dvariable& deswid, const dvariable& init, const dvariable& final)

  //ages=vector of ages, peak=ascending inflection location (as logistic), top=width of plateau, ascwid=ascent width (as log(width))

  //deswid=descent width (as log(width))

  RETURN_ARRAYS_INCREMENT();

  dvar_vector Sel_Tmp(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step1(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step2(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step3(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step4(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step5(ages.indexmin(),ages.indexmax());

  dvar_vector sel_step6(ages.indexmin(),ages.indexmax());

  dvar_vector pars_tmp(1,6); dvar_vector sel_tmp_iq(1,2);
```

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

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

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

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

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

//--------------------------------------------------------------------------------
//Spawner-recruit function (Beverton-Holt or Ricker)
FUNCTION dvariable SR_func(const dvariable& R0, const dvariable& h, const dvariable& spr_F0, const
dvariable& SSB, int func)
```

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


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

```
   break;
  }
  RETURN_ARRAYS_DECREMENT();
  return Recruits_Tmp;



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


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

```
  var=log(1.0+square(cv/wgt_dat));   // convert cv in arithmetic space to variance in log space

  LkvalTmp=sum(0.5*elem_div(square(log(elem_div((pred+small_number),(obs+small_number)))),var) );

  RETURN_ARRAYS_DECREMENT();

  return LkvalTmp;


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

//Likelihood contribution: multinomial

FUNCTION dvariable lk_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const
dvar_matrix& obs_comp, const double& ncomp, const double& minSS, const dvariable& wgt_dat)

  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp =
number of yrs in matrix, minSS=min N threshold, wgt_dat=scaling of N's

  RETURN_ARRAYS_INCREMENT();

  dvariable LkvalTmp;

  dvariable small_number=0.00001;   //KC

  LkvalTmp=0.0;

  for (int ii=1; ii<=ncomp; ii++)

  {if (nsamp(ii)>=minSS)

    {LkvalTmp-=wgt_dat*nsamp(ii)*sum(elem_prod((obs_comp(ii)+small_number),

            log(elem_div((pred_comp(ii)+small_number), (obs_comp(ii)+small_number)))));

    }

  }

  RETURN_ARRAYS_DECREMENT();

  return LkvalTmp;



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

//Likelihood contribution: multinomial

FUNCTION dvariable lk_robust_multinomial(const dvar_vector& nsamp, const dvar_matrix& pred_comp, const
dvar_matrix& obs_comp, const double& ncomp, const dvariable& mbin, const double& minSS, const dvariable&
wgt_dat)
```

```
  //nsamp=vector of N's, pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, ncomp =
number of yrs in matrix, mbin=number of bins, minSS=min N threshold, wgt_dat=scaling of N's

  RETURN_ARRAYS_INCREMENT();

  dvariable LkvalTmp;

  dvariable small_number=0.00001;   //KC

  LkvalTmp=0.0;

  dvar_matrix Eprime=elem_prod((1.0-obs_comp), obs_comp)+0.1/mbin; //E' of Francis 2011, p.1131

  dvar_vector nsamp_wgt=nsamp*wgt_dat;

  //cout<<nsamp_wgt<<endl;

  for (int ii=1; ii<=ncomp; ii++)

  {if (nsamp(ii)>=minSS)

    {LkvalTmp+= sum(0.5*log(Eprime(ii))-log(small_number+mfexp(elem_div((-square(obs_comp(ii)-
pred_comp(ii))) , (Eprime(ii)*2.0/nsamp_wgt(ii)) ))) );

    }

  }

  RETURN_ARRAYS_DECREMENT();

  return LkvalTmp;




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

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

//Likelihood contribution: priors

FUNCTION  dvariable neg_log_prior(dvariable pred, const double& prior, dvariable var, int pdf)

  //prior=prior point estimate, var=variance (if negative, treated as CV in arithmetic space), pred=predicted value,
pdf=prior type (1=none, 2=lognormal, 3=normal, 4=beta)

    dvariable LkvalTmp;

    dvariable alpha, beta, ab_iq;

    dvariable big_number=1e10;

    LkvalTmp=0.0;

    // compute generic pdf's
```

```
  switch(pdf) {
    case 1: //option to turn off prior
      LkvalTmp=0.0;
      break;
    case 2: // lognormal
      if(prior<=0.0) cout << "YIKES: Don't use a lognormal distn for a negative prior" << endl;
      else if(pred<=0) LkvalTmp=big_number=1e10;
      else {
        if(var<0.0) var=log(1.0+var*var) ;      // convert cv to variance on log scale
        LkvalTmp= 0.5*( square(log(pred/prior))/var + log(var) );
      }
    break;
    case 3: // normal
      if(var<0.0 && prior!=0.0) var=square(var*prior);       // convert cv to variance on observation scale
      else if(var<0.0 && prior==0.0) var=-var;               // cv not really appropriate if prior value equals zero
      LkvalTmp= 0.5*( square(pred-prior)/var + log(var) );
      break;
    case 4: // beta
      if(var<0.0) var=square(var*prior);          // convert cv to variance on observation scale
      if(prior<=0.0 || prior>=1.0) cout << "YIKES: Don't use a beta distn for a prior outside (0,1)" << endl;
      ab_iq=prior*(1.0-prior)/var - 1.0; alpha=prior*ab_iq; beta=(1.0-prior)*ab_iq;
      if(pred>=0 && pred<=1) LkvalTmp= (1.0-alpha)*log(pred)+(1.0-beta)*log(1.0-pred)-
gammln(alpha+beta)+gammln(alpha)+gammln(beta);
      else LkvalTmp=big_number;
      break;
    default: // no such prior pdf currently available
      cout << "The prior must be either 1(lognormal), 2(normal), or 3(beta)." << endl;
      cout << "Presently it is " << pdf << endl;
      exit(0);
```

```
  }
  return LkvalTmp;


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

```
//--------------------------------------------------------------------------------

//SDNR: lognormal likelihood

FUNCTION dvariable sdnr_lognormal(const dvar_vector& pred, const dvar_vector& obs, const dvar_vector& cv, const dvariable& wgt_dat)

  //nyr=number of years of data, pred=vector of predicted data, obs=vector of observed data, cv=vector of cv's, wgt_dat=likelihood weight for data source

  RETURN_ARRAYS_INCREMENT();

  dvariable SdnrTmp;

  dvariable small_number=0.00001;

  dvariable n;

  dvar_vector res(cv.indexmin(),cv.indexmax());

  SdnrTmp=0.0;

  res=elem_div(log(elem_div(obs+small_number,pred+small_number)),sqrt(log(1+square(cv/wgt_dat))));

  n=cv.indexmax()-cv.indexmin()+1;

  SdnrTmp=sqrt(sum(square(res-(sum(res)/n))/(n-1.0)));

  RETURN_ARRAYS_DECREMENT();

  return SdnrTmp;


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

REPORT_SECTION


  if (last_phase())

  {

     cout<<"start report"<<endl;

     get_weighted_current();

     cout<<"got weighted"<<endl;

     get_msy();

     cout<<"got msy"<<endl;

 get_per_recruit_stuff();
```

```
get_miscellaneous_stuff();
cout<<"got misc stuff"<<endl;
//cout<<"got per recruit"<<endl;
get_effective_sample_sizes();

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

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

```
    // cout << F_initial << endl;


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

     report << "N" << endl;

     report << N<<endl;

     report << "F" << endl;

     report << F <<endl;

//       report <<"lenprob" <<endl;

//       report << lenprob<<endl;


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

     sdnr_lc_mcvt=sdnr_multinomial(nyr_mcvt_lenc, lenbins, nsamp_mcvt_lenc, pred_mcvt_lenc, obs_mcvt_lenc,
w_lc_mcvt);

     sdnr_lc_HB=sdnr_multinomial(nyr_HB_lenc, lenbins, nsamp_HB_lenc, pred_HB_lenc, obs_HB_lenc,
w_lc_HB);

     sdnr_lc_HB_D=sdnr_multinomial(nyr_HB_D_lenc, lenbins, nsamp_HB_D_lenc, pred_HB_D_lenc,
obs_HB_D_lenc, w_lc_HB_D);

     sdnr_lc_GR=sdnr_multinomial(nyr_GR_lenc, lenbins, nsamp_GR_lenc, pred_GR_lenc, obs_GR_lenc,
w_lc_GR);


     sdnr_ac_cH=sdnr_multinomial(nyr_cH_agec, agebins_agec, nsamp_cH_agec, pred_cH_agec, obs_cH_agec,
w_ac_cH);

     sdnr_ac_mcvt=sdnr_multinomial(nyr_mcvt_agec, agebins_agec, nsamp_mcvt_agec, pred_mcvt_agec,
obs_mcvt_agec, w_ac_mcvt);

     sdnr_ac_HB=sdnr_multinomial(nyr_HB_agec, agebins_agec, nsamp_HB_agec, pred_HB_agec, obs_HB_agec,
w_ac_HB);

//     sdnr_ac_GR=sdnr_multinomial(nyr_GR_agec, agebins_agec, nsamp_GR_agec, pred_GR_agec, obs_GR_agec,
w_ac_GR);


//     sdnr_I_cH=sdnr_lognormal(pred_cH_cpue, obs_cH_cpue, cH_cpue_cv, w_I_cH);

     sdnr_I_mcvt=sdnr_lognormal(pred_mcvt_cpue, obs_mcvt_cpue, mcvt_cpue_cv, w_I_mcvt);

//     sdnr_I_vid=sdnr_lognormal(pred_vid_cpue, obs_vid_cpue, vid_cpue_cv, w_I_vid);

//     sdnr_I_HB=sdnr_lognormal(pred_HB_cpue, obs_HB_cpue, HB_cpue_cv, w_I_HB);
```

```
//      sdnr_I_GR=sdnr_lognormal(pred_GR_cpue, obs_GR_cpue, GR_cpue_cv, w_I_GR);



//#########################################################################################################
#####

    //##  Passing parameters to vector for bounds check plotting

//#########################################################################################################
#####

     Linf_out(8)=Linf; Linf_out(1,7)=set_Linf;

     K_out(8)=K; K_out(1,7)=set_K;

     t0_out(8)=t0; t0_out(1,7)=set_t0;

     len_cv_val_out(8)=len_cv_val; len_cv_val_out(1,7)=set_len_cv;


     Linf_L_out(8)=Linf_L; Linf_L_out(1,7)=set_Linf_L;

     K_L_out(8)=K_L; K_L_out(1,7)=set_K_L;

     t0_L_out(8)=t0_L; t0_L_out(1,7)=set_t0_L;

     len_cv_val_L_out(8)=len_cv_val_L; len_cv_val_L_out(1,7)=set_len_cv_L;


     Linf_mcvt_out(8)=Linf_mcvt; Linf_mcvt_out(1,7)=set_Linf_mcvt;

     K_mcvt_out(8)=K_mcvt; K_mcvt_out(1,7)=set_K_mcvt;

     t0_mcvt_out(8)=t0_mcvt; t0_mcvt_out(1,7)=set_t0_mcvt;

     len_cv_val_mcvt_out(8)=len_cv_val_mcvt; len_cv_val_mcvt_out(1,7)=set_len_cv_mcvt;


     log_R0_out(8)=log_R0; log_R0_out(1,7)=set_log_R0;

     M_constant_out(8)=M_constant; M_constant_out(1,7)=set_M_constant;

     steep_out(8)=steep; steep_out(1,7)=set_steep;

     rec_sigma_out(8)=rec_sigma; rec_sigma_out(1,7)=set_rec_sigma;

     R_autocorr_out(8)=R_autocorr; R_autocorr_out(1,7)=set_R_autocorr;
```

```
    selpar_L50_cH1_out(8)=selpar_L50_cH1; selpar_L50_cH1_out(1,7)=set_selpar_L50_cH1;
    selpar_slope_cH1_out(8)=selpar_slope_cH1; selpar_slope_cH1_out(1,7)=set_selpar_slope_cH1;
//    selpar_L50_cH2_out(8)=selpar_L50_cH2; selpar_L50_cH2_out(1,7)=set_selpar_L50_cH2;
//    selpar_slope_cH2_out(8)=selpar_slope_cH2; selpar_slope_cH2_out(1,7)=set_selpar_slope_cH2;
//    selpar_L50_cH3_out(8)=selpar_L50_cH3; selpar_L50_cH3_out(1,7)=set_selpar_L50_cH3;
//    selpar_slope_cH3_out(8)=selpar_slope_cH3; selpar_slope_cH3_out(1,7)=set_selpar_slope_cH3;

    selpar_L50_mcvt_out(8)=selpar_L50_mcvt; selpar_L50_mcvt_out(1,7)=set_selpar_L50_mcvt;
    selpar_slope_mcvt_out(8)=selpar_slope_mcvt; selpar_slope_mcvt_out(1,7)=set_selpar_slope_mcvt;

    selpar_L51_HB1_out(8)=selpar_L51_HB1; selpar_L51_HB1_out(1,7)=set_selpar_L51_HB1;
    selpar_slope1_HB1_out(8)=selpar_slope1_HB1; selpar_slope1_HB1_out(1,7)=set_selpar_slope1_HB1;
    selpar_L52_HB1_out(8)=selpar_L52_HB1; selpar_L52_HB1_out(1,7)=set_selpar_L52_HB1;
    selpar_slope2_HB1_out(8)=selpar_slope2_HB1; selpar_slope2_HB1_out(1,7)=set_selpar_slope2_HB1;

    selpar_L50_HB2_out(8)=selpar_L50_HB2; selpar_L50_HB2_out(1,7)=set_selpar_L50_HB2;
    selpar_slope_HB2_out(8)=selpar_slope_HB2; selpar_slope_HB2_out(1,7)=set_selpar_slope_HB2;
    selpar_afull_HB2_out(8)=selpar_afull_HB2; selpar_afull_HB2_out(1,7)=set_selpar_afull_HB2;
    selpar_sigma_HB2_out(8)=selpar_sigma_HB2; selpar_sigma_HB2_out(1,7)=set_selpar_sigma_HB2;

//    selpar_L50_HB3_out(8)=selpar_L50_HB3; selpar_L50_HB3_out(1,7)=set_selpar_L50_HB3;
//    selpar_slope_HB3_out(8)=selpar_slope_HB3; selpar_slope_HB3_out(1,7)=set_selpar_slope_HB3;

    selpar_L50_HB_D_out(8)=selpar_L50_HB_D; selpar_L50_HB_D_out(1,7)=set_selpar_L50_HB_D;
    selpar_slope_HB_D_out(8)=selpar_slope_HB_D; selpar_slope_HB_D_out(1,7)=set_selpar_slope_HB_D;
    selpar_afull_HB_D_out(8)=selpar_afull_HB_D; selpar_afull_HB_D_out(1,7)=set_selpar_afull_HB_D;
    selpar_sigma_HB_D_out(8)=selpar_sigma_HB_D; selpar_sigma_HB_D_out(1,7)=set_selpar_sigma_HB_D;
//
    selpar_L51_GR_out(8)=selpar_L51_GR; selpar_L51_GR_out(1,7)=set_selpar_L51_GR;
```

```
      selpar_slope1_GR_out(8)=selpar_slope1_GR; selpar_slope1_GR_out(1,7)=set_selpar_slope1_GR;
      selpar_L52_GR_out(8)=selpar_L52_GR; selpar_L52_GR_out(1,7)=set_selpar_L52_GR;
      selpar_slope2_GR_out(8)=selpar_slope2_GR; selpar_slope2_GR_out(1,7)=set_selpar_slope2_GR;

      log_q_cH_out(8)=log_q_cH; log_q_cH_out(1,7)=set_log_q_cH;
      log_q_mcvt_out(8)=log_q_mcvt; log_q_mcvt_out(1,7)=set_log_q_mcvt;
//      log_q_vid_out(8)=log_q_vid; log_q_vid_out(1,7)=set_log_q_vid;
      log_q_HB_out(8)=log_q_HB; log_q_HB_out(1,7)=set_log_q_HB;
      log_q_GR_out(8)=log_q_GR; log_q_GR_out(1,7)=set_log_q_GR;

      log_avg_F_cH_out(8)=log_avg_F_cH; log_avg_F_cH_out(1,7)=set_log_avg_F_cH;
      log_avg_F_HB_out(8)=log_avg_F_HB; log_avg_F_HB_out(1,7)=set_log_avg_F_HB;
      log_avg_F_GR_out(8)=log_avg_F_GR; log_avg_F_GR_out(1,7)=set_log_avg_F_GR;
      log_avg_F_HB_D_out(8)=log_avg_F_HB_D; log_avg_F_HB_D_out(1,7)=set_log_avg_F_HB_D;
      log_avg_F_GR_D_out(8)=log_avg_F_GR_D; log_avg_F_GR_D_out(1,7)=set_log_avg_F_GR_D;
      F_init_out(8)=F_init; F_init_out(1,7)=set_F_init;

      log_rec_dev_out(styr_rec_dev, endyr_rec_dev)=log_rec_dev;
      log_F_dev_cH_out(styr_cH_L,endyr_cH_L)=log_F_dev_cH;
      log_F_dev_HB_out(styr_HB_L,endyr_HB_L)=log_F_dev_HB;
      log_F_dev_GR_out(styr_GR_L,endyr_GR_L)=log_F_dev_GR;
      log_F_dev_HB_D_out(styr_HB_D,endyr_HB_D)=log_F_dev_HB_D;
      log_F_dev_GR_D_out(styr_GR_D,endyr_GR_D)=log_F_dev_GR_D;

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

  } //endl last phase loop
```

ADMB Dat file for the BAM:

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

##

Data Input File

SEDAR 41 gray triggerfish benchmark assessment Oct 2015

##

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

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

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

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

#Starting and ending year of model

1988

2014

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

1988

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

2014

#3 phases of constraints on recruitment deviations:

#allows possible heavier constraint (weights defined later) in early and late period, with lighter constraint in the middle

#ending years of recruitment constraint phases

1988

2011

#Ending year for selectivity blocks (no selectivity blocks for gt base model)

2014

#Number of ages in population model(8 classes are 1,...,N+) //assumes last age is plus group

8

#Vector of agebins, last is a plus group

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0

#Number of ages used to match age comps: first age must be same as popn, plus group may differ

8

#Vector of agebins, last is a plus group

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0

#Number length bins used to match length comps and width of bins

19 #number bins

30.0 #width of bins (mm)

#Vector of length bins (mm)(midpoint of bin) used to match length comps and bins used to compute plus group

150.0 180.0 210.0 240.0 270.0 300.0 330.0 360.0 390.0 420.0 450.0 480.0 510.0
540.0 570.0 600.0 630.0 660.0 690.0

#Max value of F used in spr and msy calculations

3.0

#Number of iterations in spr calculations

10001

#Number years at end of time series over which to average sector Fs, for weighted selectivities

3

#Multiplicative bias correction of recruitment (may set to 1.0 for none or negative to compute from recruitment variance)

-1.0

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

##-- BAM DATA SECTION: observed data section

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

##Commercial Handline (cH)
###

#Comm handline logbook CPUE index

#Starting and ending years of CPUE index

1993

2009

#Observed CPUE and CVs

0.76 0.89 1.01 1.04 1.53 1.38 1.06 0.76 0.69 0.66 0.75 1.14 1.24 0.99 1.00 0.98 1.13

#CV on FD indices (changed from index estimated CV to 0.2)

0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2

#0.07 0.05 0.05 0.05 0.04 0.05 0.05 0.05 0.05 0.05 0.06 0.05 0.05 0.05 0.05 0.05 0.05

#Starting and ending years for commercial fishery landings (handline+other+discards) time series

1988

2014

#Commercial handline landings vector (1000 lb whole weight) and assumed CVs (handline + other + discards)

#Dead discards are included from 1988-2014 (based on observed values 1993-2014 and predicted values 1988-1992)

79.832 96.487 194.797 271.694 263.032 329.064 410.679 488.437 441.970 536.451 424.389 279.378 196.777 215.776 204.482 192.111 252.412 276.017 241.487 323.169 320.128 370.283 453.214 498.734 311.367 339.273 284.758

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05

#Number and vector of years of commercial handline length compositions (weighted)

27

1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
2013 2014

#Sample size of length composition data (first row observed Ntrips, second row Nfish; weighted)

32.0 38.0 38.0 36.0 34.0 77.0 66.0 136.0 78.0 51.0 65.0 95.0 154.0
137.0 93.0 78.0 148.0 136.0 220.0 285.0 269.0 242.0 269.0 297.0 202.0
148.0 51.0

212.0 339.0 650.0 572.0 487.0 1107.0 1480.0 3131.0 1789.0 906.0 1338.0 1822.0 2565.0
2020.0 1429.0 2230.0 3478.0 2678.0 2706.0 2130.0 1583.0 1831.0 2447.0 3162.0 1992.0
1594.0 406.0

#Commercial handline length compositions (3 cm length bins, weighted)

0.0000 0.0000 0.0042 0.0043 0.0151 0.0255 0.0453 0.1109 0.1669 0.2117 0.1699 0.0846 0.0748
0.0605 0.0212 0.0050 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0028 0.0222 0.0660 0.1158 0.1914 0.2159 0.1962 0.1114 0.0344
0.0357 0.0054 0.0028 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0013 0.0029 0.0143 0.0533 0.1900 0.2526 0.2122 0.1599 0.0589 0.0299
0.0155 0.0092 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0028 0.0059 0.0440 0.1098 0.1923 0.2077 0.2235 0.1354 0.0615 0.0142
0.0000 0.0028 0.0000 0.0000 0.0000 0.0000

0.0015 0.0000 0.0000 0.0000 0.0073 0.0589 0.1742 0.1994 0.2192 0.1523 0.0990 0.0624 0.0183
0.0042 0.0014 0.0018 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0039 0.0347 0.1368 0.2349 0.2201 0.1628 0.1039 0.0573 0.0297
0.0115 0.0008 0.0009 0.0008 0.0000 0.0018

0.0000 0.0000 0.0000 0.0000 0.0030 0.0357 0.1617 0.2455 0.2162 0.1725 0.0832 0.0512 0.0200
0.0088 0.0022 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0032 0.0298 0.0859 0.1463 0.2183 0.2236 0.1517 0.0859 0.0324 0.0132
0.0073 0.0019 0.0000 0.0000 0.0003 0.0000

0.0000 0.0000 0.0000 0.0052 0.0223 0.1047 0.2287 0.2656 0.1779 0.1022 0.0558 0.0249 0.0061
0.0033 0.0021 0.0008 0.0004 0.0000 0.0000

0.0000 0.0000 0.0000 0.0129 0.0271 0.1190 0.2036 0.2795 0.1696 0.0929 0.0498 0.0292 0.0109
0.0055 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0023 0.0098 0.0401 0.0984 0.2200 0.2502 0.2109 0.1092 0.0415 0.0106 0.0049
0.0013 0.0007 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0006 0.0182 0.0589 0.1632 0.2416 0.1965 0.1683 0.0960 0.0318 0.0091
0.0047 0.0074 0.0029 0.0008 0.0000 0.0000

0.0000 0.0000 0.0021 0.0077 0.0340 0.0939 0.1537 0.2175 0.2116 0.1391 0.0902 0.0356 0.0111
0.0033 0.0002 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0006 0.0036 0.0281 0.0895 0.2017 0.2345 0.2091 0.1219 0.0709 0.0258 0.0092
0.0031 0.0004 0.0000 0.0000 0.0000 0.0014

0.0000 0.0000 0.0000 0.0042 0.0409 0.0954 0.1910 0.2186 0.2130 0.1370 0.0620 0.0292 0.0061
0.0016 0.0000 0.0004 0.0000 0.0000 0.0005

0.0000 0.0000 0.0015 0.0007 0.0144 0.0829 0.2051 0.2549 0.2311 0.1136 0.0590 0.0275 0.0077
0.0007 0.0005 0.0000 0.0000 0.0000 0.0004

0.0000 0.0000 0.0003 0.0020 0.0140 0.0828 0.2025 0.3032 0.2256 0.1010 0.0470 0.0134 0.0062
0.0017 0.0000 0.0002 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0015 0.0101 0.0670 0.1638 0.2604 0.2663 0.1363 0.0645 0.0188 0.0096
0.0010 0.0007 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0003 0.0036 0.0142 0.0673 0.1511 0.2167 0.2294 0.1759 0.0908 0.0352 0.0132
0.0015 0.0000 0.0003 0.0000 0.0000 0.0003

0.0000 0.0000 0.0000 0.0009 0.0126 0.0491 0.1130 0.2032 0.2386 0.1811 0.1207 0.0605 0.0136
0.0048 0.0005 0.0008 0.0006 0.0000 0.0000

0.0000 0.0000 0.0000 0.0006 0.0063 0.0261 0.1163 0.1756 0.2823 0.1878 0.1041 0.0756 0.0226
0.0027 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0005 0.0023 0.0219 0.0647 0.1785 0.2082 0.2264 0.1403 0.0867 0.0363 0.0249
0.0072 0.0014 0.0000 0.0009 0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0094 0.0511 0.1289 0.1983 0.2329 0.1737 0.1112 0.0654 0.0235
0.0048 0.0005 0.0002 0.0000 0.0000 0.0000

0.0000 0.0000 0.0002 0.0007 0.0080 0.0370 0.1331 0.2117 0.2198 0.1838 0.1113 0.0601 0.0299
0.0045 0.0001 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0017 0.0097 0.0412 0.1037 0.1854 0.1993 0.1833 0.1476 0.0779 0.0437
0.0036 0.0016 0.0005 0.0000 0.0000 0.0008

0.0000 0.0000 0.0000 0.0010 0.0099 0.0612 0.1361 0.2022 0.1701 0.1692 0.1262 0.0832 0.0273
0.0131 0.0002 0.0002 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0042 0.0064 0.0415 0.0891 0.1756 0.1857 0.2207 0.1260 0.1072 0.0396
0.0019 0.0022 0.0000 0.0000 0.0000 0.0000

#Number and vector of years of age compositions for commercial handline fleet

11

2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#Sample size of age composition data (first row observed Ntrips, second row Nfish)

25.0 47.0 86.0 196.0 205.0 180.0 215.0 211.0 110.0 97.0 69.0

188.0 386.0 463.0 681.0 736.0 686.0 965.0 1237.0 756.0 563.0 431.0

#Commercial handline age compositions (weighted)

0.0000 0.0056 0.1796 0.3708 0.2900 0.1258 0.0209 0.0073

0.0046 0.0567 0.1875 0.3090 0.2506 0.1210 0.0477 0.0228

0.0025 0.0447 0.1799 0.2604 0.2589 0.1565 0.0723 0.0248

0.0172 0.0629 0.1826 0.2660 0.2227 0.1382 0.0707 0.0397

0.0014 0.0244 0.1320 0.2679 0.2319 0.1547 0.1079 0.0798

0.0000 0.0219 0.1844 0.2542 0.2498 0.1537 0.0744 0.0616

0.0020 0.0371 0.1362 0.2474 0.2240 0.1644 0.1047 0.0844

0.0000 0.0358 0.1631 0.2924 0.2570 0.1438 0.0551 0.0528

0.0028 0.0404 0.1453 0.2646 0.2360 0.1821 0.0760 0.0527

0.0207 0.2280 0.2081 0.2036 0.1808 0.0999 0.0407 0.0182

0.0018 0.0658 0.2556 0.3207 0.1785 0.1091 0.0372 0.0312

###SERFS Chevron Trap (mcvt)
###

1990

2014

#Observed CPUE (numbers) and CV vectors, respectively

#Combined trap-video index (based on Conn method)

0.283 1.079 0.865 0.801 1.030 1.332 1.583 1.443 1.701 0.752 0.649 0.882 1.498
0.833 1.267 0.766 0.556 0.949 0.894 0.701 0.671 0.870 1.063 1.238 1.294

0.320 0.235 0.258 0.239 0.228 0.220 0.217 0.224 0.227 0.269 0.282 0.248 0.242
0.310 0.240 0.249 0.267 0.253 0.254 0.260 0.254 0.187 0.177 0.167 0.198

#Below is for trap only (no video)

#0.25 1.08 0.85 0.79 1.03 1.35 1.62 1.47 1.75 0.73 0.62 0.87 1.53
0.80 1.28 0.75 0.53 0.94 0.88 0.68 0.65 0.82 1.02 1.25 1.49

#0.20 0.12 0.16 0.11 0.01 0.10 0.10 0.11 0.13 0.17 0.19 0.13 0.15
0.25 0.14 0.13 0.15 0.15 0.15 0.15 0.13 0.11 0.09 0.08 0.08

#Number and vector of years of SERFS trap length compositions

25

1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#Sample size of length composition data (first row observed Ntrips, second row Nfish)

41.0 134.0 88.0 118.0 154.0 156.0 179.0 194.0 124.0 62.0 92.0 99.0 112.0 34.0 96.0 108.0
75.0 123.0 72.0 90.0 216.0 169.0 341.0 367.0 464.0

78.0 398.0 204.0 298.0 447.0 669.0 1198.0 958.0 519.0 190.0 269.0 258.0 348.0 67.0 262.0 382.0
174.0 423.0 334.0 319.0 586.0 595.0 1148.0 1270.0 1658.0

#SERFS trap length compositions (3 cm length bins)

0.0512 0.0769 0.1795 0.1154 0.0384 0.1026 0.0897 0.1154 0.1282 0.0769 0.0128 0.0000 0.0000
0.0000 0.0128 0.0000 0.0000 0.0000 0.0000

0.0979 0.2462 0.1584 0.1131 0.1006 0.0955 0.0478 0.0351 0.0328 0.0327 0.0302 0.0050 0.0025
0.0025 0.0000 0.0000 0.0000 0.0000 0.0000

0.0245 0.0539 0.1029 0.1127 0.1127 0.1666 0.0980 0.0931 0.0784 0.0686 0.0490 0.0245 0.0000
0.0098 0.0049 0.0000 0.0000 0.0000 0.0000

0.0303 0.0637 0.1040 0.2080 0.1913 0.1611 0.0739 0.0638 0.0370 0.0336 0.0101 0.0134 0.0101
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0067 0.0448 0.0626 0.0872 0.1096 0.1634 0.1455 0.1297 0.1342 0.0671 0.0335 0.0089 0.0045
0.0000 0.0022 0.0000 0.0000 0.0000 0.0000

0.0284 0.0537 0.0703 0.1076 0.0538 0.0672 0.1121 0.1749 0.1689 0.0897 0.0583 0.0120 0.0030
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0116 0.0376 0.0442 0.0634 0.1845 0.2747 0.2170 0.0868 0.0417 0.0268 0.0076 0.0025 0.0000
0.0016 0.0000 0.0000 0.0000 0.0000 0.0000

0.0031 0.0251 0.0407 0.0595 0.0679 0.1753 0.2578 0.2306 0.0804 0.0386 0.0146 0.0052 0.0010
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0077 0.0116 0.0193 0.0597 0.1002 0.1657 0.2351 0.2389 0.1021 0.0482 0.0078 0.0039 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0105 0.0106 0.1000 0.0843 0.1158 0.1684 0.1368 0.1158 0.1632 0.0579 0.0263 0.0106 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0186 0.0261 0.0446 0.0632 0.1376 0.2119 0.2900 0.1301 0.0632 0.0111 0.0037 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0078 0.0233 0.1008 0.0814 0.1008 0.0853 0.1976 0.2054 0.0930 0.0621 0.0349 0.0078 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0029 0.0603 0.1265 0.1264 0.0834 0.1093 0.1264 0.1264 0.1120 0.0862 0.0345 0.0029 0.0029
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0298 0.0448 0.0448 0.0447 0.2090 0.1941 0.1791 0.1344 0.1045 0.0149 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0153 0.0382 0.0535 0.1030 0.1641 0.2138 0.1985 0.1069 0.0649 0.0228 0.0152 0.0038 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0026 0.0104 0.0130 0.0445 0.0603 0.0969 0.1963 0.2199 0.1963 0.1361 0.0210 0.0026 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0287 0.0402 0.0690 0.0862 0.1035 0.1437 0.1379 0.1264 0.1379 0.0804 0.0344 0.0114 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0142 0.0426 0.0543 0.1111 0.1678 0.2057 0.1631 0.1348 0.0733 0.0260 0.0071 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0090 0.0330 0.0749 0.0270 0.0659 0.0838 0.1766 0.1796 0.2036 0.0898 0.0450 0.0090 0.0030
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0219 0.0407 0.1411 0.2696 0.2477 0.1254 0.0941 0.0345 0.0251 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0017 0.0000 0.0136 0.0376 0.0922 0.1655 0.2167 0.1911 0.1434 0.0785 0.0358 0.0170 0.0034
0.0017 0.0000 0.0017 0.0000 0.0000 0.0000

0.0051 0.0068 0.0286 0.0403 0.0941 0.1328 0.1865 0.2185 0.1513 0.0857 0.0353 0.0117 0.0034
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0131 0.0261 0.0653 0.1672 0.1498 0.1394 0.1297 0.1620 0.0976 0.0374 0.0052 0.0061
0.0009 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0094 0.0307 0.0480 0.0953 0.1812 0.2725 0.1582 0.1015 0.0582 0.0244 0.0134 0.0055
0.0016 0.0000 0.0000 0.0000 0.0000 0.0000

0.0012 0.0084 0.0235 0.0573 0.0814 0.1882 0.2449 0.1943 0.0911 0.0826 0.0204 0.0048 0.0000
0.0018 0.0000 0.0000 0.0000 0.0000 0.0000

#Number and vector of years of age compositions for SERFS chevron trap

24

1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#Sample size of age composition data (first row observed Ntrips, second row Nfish)

47.0 70.0 112.0 142.0 134.0 166.0 164.0 118.0 60.0 86.0 78.0 102.0 33.0 74.0 99.0 64.0 96.0 64.0 79.0 97.0 116.0 190.0 281.0 304.0

389.0 209.0 308.0 465.0 686.0 1219.0 963.0 530.0 202.0 268.0 273.0 361.0 74.0 212.0 358.0
186.0 352.0 272.0 238.0 197.0 338.0 450.0 909.0 976.0

#SERFS chevron trap age compositions

0.5362 0.2772 0.0648 0.0699 0.0208 0.0155 0.0104 0.0051

0.2277 0.3069 0.1980 0.1831 0.0495 0.0248 0.0099 0.0000

0.1395 0.3289 0.2991 0.1129 0.0930 0.0199 0.0066 0.0000

0.1538 0.2565 0.2992 0.1752 0.0727 0.0192 0.0213 0.0021

0.1016 0.2351 0.2946 0.2017 0.1089 0.0174 0.0232 0.0175

0.0665 0.1980 0.2893 0.2506 0.1233 0.0362 0.0246 0.0114

0.0585 0.1743 0.3027 0.2537 0.1367 0.0376 0.0230 0.0136

0.0266 0.1121 0.3061 0.3061 0.1749 0.0361 0.0323 0.0057

0.1238 0.1980 0.2524 0.1931 0.1485 0.0545 0.0248 0.0050

0.0909 0.1705 0.3068 0.2235 0.1250 0.0455 0.0189 0.0190

0.2333 0.2148 0.2370 0.2037 0.0778 0.0037 0.0222 0.0074

0.1923 0.3022 0.1676 0.1401 0.1044 0.0412 0.0412 0.0109

0.0811 0.1757 0.2973 0.1622 0.2297 0.0270 0.0270 0.0000

0.1925 0.2864 0.3005 0.1314 0.0469 0.0141 0.0281 0.0000

0.0167 0.0866 0.2207 0.3547 0.1732 0.0782 0.0530 0.0168

0.0699 0.1290 0.3011 0.1774 0.2097 0.0591 0.0323 0.0216

0.0426 0.2017 0.2756 0.2472 0.1335 0.0511 0.0284 0.0199

0.1115 0.1730 0.2577 0.1846 0.1346 0.0500 0.0461 0.0424

0.1815 0.3840 0.2700 0.1054 0.0337 0.0211 0.0000 0.0042

0.1701 0.3557 0.2731 0.1186 0.0618 0.0000 0.0052 0.0155

0.1098 0.1751 0.2789 0.1928 0.1365 0.0534 0.0238 0.0297

0.0468 0.1715 0.2249 0.2517 0.1871 0.0401 0.0289 0.0489

0.1089 0.2967 0.2956 0.1367 0.1000 0.0256 0.0200 0.0167

0.1175 0.2506 0.2746 0.1709 0.0943 0.0514 0.0262 0.0146

##SEFIS Video Index (vid)
###

#2011

#2014

#Observed CPUE (numbers) and CV vectors, respectively

#0.88 1.07 1.14 0.91

#0.12 0.11 0.09 0.10

##########headboat (HB)
##########

#Headboat Index

#Starting and ending years of CPUE index

1995

2009

#Observed CPUE and CVs

0.88 0.94 1.22 1.00 0.87 0.59 0.60 0.73 0.93 1.52 1.19 0.97 1.11
1.06 1.40

#CV on FD indices (changed from index standardization values to 0.2)

0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2

#0.04 0.04 0.04 0.03 0.03 0.04 0.04 0.04 0.04 0.03 0.04 0.04 0.03
0.03 0.03

#Headboat landings

#Starting and ending years of HB landings time series, respectively

1988

2014

#Observed Headboat landings (1000s of fish) and assumed CVs

34.926 37.367 71.704 85.529 91.733 107.070 90.387 93.367 89.954 106.17 65.857 37.218 34.092
32.978 57.63 45.751 78.073 63.582 43.151 66.403 44.758 59.945 68.807 53.356 49.096 56.487
53.108

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05

#Number and vector of years of length compositions for Headboat fleet

27

1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
2013 2014

#Sample size of HB length composition data by year (first row observed Ntrips, second row Nfish; weighted, includes SRHS and Va north)

228.0 265.0 252.0 196.0 243.0 269.0 244.0 250.0 192.0 312.0 348.0 265.0 189.0
207.0 249.0 306.0 331.0 260.0 273.0 342.0 200.0 316.0 335.0 327.0 313.0
401.0 394.0

431.0 710.0 794.0 597.0 740.0 812.0 1030.0 871.0 1024.0 1568.0 954.0 714.0 418.0
496.0 620.0 969.0 1272.0 956.0 772.0 1007.0 673.0 930.0 1324.0 1087.0 1265.0
2523.0 1699.0

#HB length composition samples (year,3 cm lengthbin; weighted, includes SRHS and Va north)

0.00000 0.00334 0.01758 0.06776 0.22333 0.23175 0.15505 0.10394 0.07210 0.05614 0.02744 0.01112 0.01274
0.01684 0.00000 0.00000 0.00000 0.00000 0.00086

0.00000 0.00116 0.01503 0.09250 0.19649 0.17442 0.10258 0.10347 0.10900 0.07697 0.05041 0.02868 0.02637
0.01263 0.00229 0.00458 0.00229 0.00000 0.00116

0.00000 0.00138 0.01516 0.07262 0.16709 0.22851 0.17306 0.16397 0.09029 0.04889 0.02170 0.00829 0.00808
0.00099 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00494 0.04055 0.12496 0.18085 0.14431 0.16920 0.14954 0.07173 0.05814 0.02613 0.01543
0.00710 0.00474 0.00000 0.00000 0.00000 0.00237

0.00000 0.00114 0.00604 0.04352 0.11185 0.18751 0.26355 0.19470 0.09569 0.05154 0.02332 0.01263 0.00706
0.00148 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00266 0.02167 0.05520 0.14144 0.28127 0.26416 0.11837 0.05300 0.03582 0.01263 0.00792
0.00292 0.00293 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00175 0.02089 0.06032 0.15073 0.27281 0.24967 0.14770 0.06480 0.01935 0.00811 0.00212
0.00106 0.00000 0.00069 0.00000 0.00000 0.00000

0.00000 0.00000 0.00866 0.02331 0.07644 0.20616 0.27311 0.18031 0.12514 0.06117 0.03595 0.00711 0.00070
0.00125 0.00070 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00196 0.01070 0.07020 0.25782 0.34982 0.17973 0.07807 0.03318 0.01073 0.00196 0.00194
0.00196 0.00196 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00807 0.07448 0.24797 0.34193 0.19285 0.08711 0.03324 0.00592 0.00307 0.00313
0.00148 0.00000 0.00000 0.00000 0.00074 0.00000

0.00124 0.00000 0.00188 0.01155 0.07754 0.20061 0.27681 0.21581 0.14033 0.06014 0.00813 0.00219 0.00031
0.00157 0.00188 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01363 0.07974 0.17443 0.29192 0.18227 0.14146 0.08606 0.01888 0.00475 0.00687
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00087 0.02601 0.14277 0.23642 0.22461 0.21281 0.09321 0.05115 0.01041 0.00174 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00087 0.03330 0.08990 0.25343 0.30671 0.19342 0.08381 0.02959 0.00724 0.00087 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00087

0.00000 0.00000 0.00256 0.05093 0.16423 0.22669 0.19046 0.22486 0.09727 0.03161 0.00381 0.00252 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00504

0.00000 0.00000 0.00000 0.02246 0.14435 0.24007 0.32657 0.18314 0.04866 0.01825 0.00825 0.00586 0.00239
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00216 0.01509 0.12317 0.27059 0.31787 0.16253 0.07551 0.02373 0.00936 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01084 0.15096 0.25489 0.19617 0.22281 0.12093 0.02517 0.01349 0.00337 0.00045
0.00045 0.00045 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00337 0.00944 0.11874 0.28407 0.24124 0.17531 0.10910 0.04365 0.00782 0.00674 0.00054
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00772 0.04991 0.22441 0.26059 0.28232 0.12432 0.03107 0.01131 0.00555 0.00020
0.00000 0.00000 0.00000 0.00257 0.00000 0.00000

0.00000 0.00000 0.00000 0.00077 0.07116 0.18394 0.25817 0.30643 0.13837 0.03245 0.00319 0.00550 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00435 0.04356 0.17369 0.38062 0.26777 0.09067 0.02138 0.01164 0.00389 0.00242
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00823 0.03762 0.18163 0.33613 0.27716 0.09514 0.04308 0.01452 0.00429 0.00179
0.00000 0.00000 0.00000 0.00000 0.00000 0.00042

0.00000 0.00000 0.00049 0.00213 0.03230 0.12389 0.29670 0.26997 0.15637 0.09491 0.01474 0.00704 0.00098
0.00049 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00183 0.00707 0.04746 0.20329 0.23090 0.20561 0.16092 0.09722 0.03005 0.01140 0.00267
0.00067 0.00000 0.00000 0.00000 0.00000 0.00092

0.00000 0.00000 0.00000 0.00281 0.04602 0.19438 0.28838 0.27459 0.12395 0.05080 0.01618 0.00214 0.00043
0.00032 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00262 0.03729 0.19826 0.29849 0.25215 0.13645 0.04196 0.02628 0.00535 0.00117
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

#Number and vector of years of age compositions for headboat fleet (weighted)

13

1990 1991 2003 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#Sample sizes of age compositions by year (first row observed Ntrips, second row Nfish; weighted)

10.0 24.0 20.0 19.0 30.0 47.0 13.0 30.0 55.0 36.0 35.0 129.0 187.0

18.0 37.0 37.0 67.0 83.0 62.0 18.0 29.0 97.0 60.0 123.0 489.0 557.0

#Age composition samples (year,age) from HB fleet (weighted)

0.00000 0.42544 0.42719 0.05789 0.08684 0.00000 0.00263 0.00000

0.00000 0.01047 0.16928 0.51134 0.19023 0.05846 0.04974 0.01047

0.00000 0.15737 0.53373 0.25020 0.05737 0.00000 0.00133 0.00000

0.00000 0.32618 0.40701 0.20260 0.05020 0.01231 0.00169 0.00000

0.09906 0.16949 0.26422 0.22761 0.21106 0.01566 0.01180 0.00109

0.01529 0.10762 0.21261 0.36255 0.23603 0.05979 0.00612 0.00000

0.00000 0.05670 0.38230 0.40206 0.12285 0.03608 0.00000 0.00000

0.00000 0.03987 0.17288 0.35065 0.41046 0.02288 0.00000 0.00327

0.00000 0.06728 0.33165 0.23863 0.27000 0.07984 0.01261 0.00000

0.00000 0.08770 0.11896 0.39726 0.22848 0.14417 0.01186 0.01157

0.09312 0.20715 0.41424 0.18013 0.05932 0.04078 0.00526 0.00000

0.07656 0.40291 0.33891 0.14390 0.02450 0.01005 0.00228 0.00090

0.08034 0.29400 0.33718 0.18803 0.07469 0.02042 0.00381 0.00154

#Headboat discards

#Starting and ending years of Headboat discards time series, respectively

1988

2014

#Observed Headboat discards (1000s of fish) and assumed CVs

1.320 1.582 0.571 0.130 0.388 0.193 0.191 0.938 0.222 0.702 1.122 2.664 3.360
3.503 6.032 2.349 1.967 2.073 2.901 1.799 10.720 10.353 19.359 11.522 12.652
8.761 12.151

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05

#Number and vector of years of Headboat discard length compositions

10

2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#sample sizes of Headboat discard length compositions by year (first row number of trip, second row number of fish)

42.0 43.0 42.0 30.0 51.0 49.0 32.0 36.0 43.0 49.0

108.0 77.0 86.0 92.0 129.0 90.0 43.0 49.0 135.0 212.0

#Headboat discard length composition by year (3cm length bins)

0.0000 0.0188 0.0512 0.4431 0.4556 0.0084 0.0136 0.0094 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0104 0.0000 0.0623 0.3174 0.4253 0.1639 0.0104 0.0000 0.0104 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0345 0.2533 0.4147 0.2747 0.0129 0.0099 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0026 0.0000 0.0144 0.1723 0.5314 0.2308 0.0223 0.0209 0.0039 0.0000 0.0013 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0262 0.0485 0.1593 0.4474 0.2923 0.0175 0.0000 0.0087 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0177 0.0177 0.2002 0.4776 0.2382 0.0309 0.0089 0.0089 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0495 0.0990 0.4181 0.3191 0.0662 0.0241 0.0241 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0528 0.2701 0.4308 0.2111 0.0176 0.0000 0.0176 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0292 0.2537 0.3878 0.3221 0.0073 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0049 0.0097 0.0243 0.1655 0.4443 0.2978 0.0438 0.0097 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

##########################General recreational (MRFSS/MRIP; GR)
##

#MRFSS Index (GR)

#Starting and ending years of CPUE index

1993

2009

#Observed CPUE and CVs

0.55 0.70 0.71 0.94 1.38 0.74 0.84 0.48 0.71 1.05 1.09 1.61 1.16
0.89 1.28 0.96 1.32

#CV on FD indices changed to 0.2 (CVs from index standardization below)

0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2

#0.13 0.15 0.15 0.14 0.14 0.12 0.14 0.12 0.10 0.09 0.08 0.09 0.09
0.09 0.10 0.10 0.09

#General Recreational landings

#Starting and ending years of landings time series, respectively

1988

2014

#Observed general recreational landings (1000s of fish) and assumed CVs; the first 3 values are estimated as (historical rec - observed HB)

113.352 394.704 238.471 313.797 192.245 268.924 171.182 148.134 275.301 290.987 93.577 112.743
107.727 126.007 184.651 170.767 217.963 207.746 169.506 308.856 429.321 497.253 325.334
167.214 264.993 166.817 230.238

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05

#Number and vector of years of MRIP length compositions for general recreational fleet

26

1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001
2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
2014

#Sample size of MRIP length composition data by year (first row observed Ntrips, second row Nfish; weighted)

30.0 27.0 31.0 44.0 44.0 53.0 42.0 57.0 35.0 24.0 77.0 16.0 67.0
107.0 99.0 127.0 76.0 86.0 105.0 99.0 124.0 144.0 77.0 99.0 105.0
165.0

85.0 62.0 67.0 91.0 128.0 264.0 125.0 263.0 140.0 71.0 188.0 60.0 150.0
266.0 234.0 382.0 193.0 159.0 290.0 208.0 341.0 592.0 430.0 298.0 516.0
495.0

#MRIP length composition samples (year,3 cm lengthbin, weighted)

0.00000 0.00000 0.00000 0.00000 0.03571 0.08929 0.07143 0.35714 0.16071 0.16071 0.01786 0.03571 0.07143
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.01961 0.05882 0.01961 0.09804 0.15686 0.19608 0.17647 0.13725 0.05882 0.05882 0.01961
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01887 0.03774 0.09434 0.22642 0.26415 0.24528 0.00000 0.03774 0.05660 0.01887
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.01333 0.00000 0.09333 0.28000 0.26667 0.14667 0.08000 0.05333 0.02667 0.00000 0.01333
0.01333 0.00000 0.01333 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.02679 0.06250 0.33036 0.17857 0.26786 0.07143 0.03571 0.00000 0.01786 0.00000
0.00893 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00398 0.00797 0.00398 0.06773 0.27490 0.23904 0.18327 0.15538 0.04382 0.00797 0.00398
0.00797 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.06195 0.21239 0.23894 0.24779 0.15929 0.06195 0.00885 0.00885 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01938 0.09302 0.14729 0.12016 0.27132 0.17829 0.10078 0.05426 0.01550 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00833 0.00833 0.03333 0.16667 0.45000 0.15000 0.12500 0.02500 0.01667 0.00833 0.00000
0.00833 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.03390 0.05085 0.35593 0.33898 0.08475 0.11864 0.01695 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00310 0.03041 0.07263 0.17877 0.23072 0.15442 0.13767 0.09797 0.05335 0.00929 0.01056
0.02111 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.02632 0.05263 0.07895 0.07895 0.21053 0.34211 0.13158 0.02632 0.05263 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00000 0.05254 0.11594 0.39761 0.23726 0.10131 0.05417 0.02004 0.00704 0.00704
0.00704 0.00000 0.00000 0.00000 0.00000 0.00000

0.00247 0.00000 0.00000 0.00787 0.09245 0.23677 0.25251 0.16885 0.11020 0.06884 0.02902 0.02361 0.00000
0.00494 0.00000 0.00000 0.00247 0.00000 0.00000

0.00000 0.00000 0.00000 0.01740 0.09709 0.26581 0.31992 0.11188 0.07376 0.06038 0.04227 0.01148 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.02475 0.08471 0.22986 0.30597 0.17978 0.09302 0.04847 0.01261 0.01504 0.00290
0.00290 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00499 0.00000 0.14429 0.18497 0.21719 0.22485 0.14044 0.06255 0.02072 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01056 0.09687 0.20220 0.17702 0.22452 0.14964 0.07169 0.03057 0.01583 0.01583
0.00528 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00706 0.05161 0.16987 0.30932 0.24789 0.10379 0.06907 0.03044 0.01095 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01560 0.06534 0.20905 0.23732 0.19441 0.16775 0.05380 0.03934 0.01333 0.00406
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00228 0.03188 0.05008 0.19085 0.31605 0.21133 0.11120 0.05450 0.02276 0.00906 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00236 0.00000 0.00380 0.04698 0.16961 0.29287 0.25123 0.12366 0.06353 0.02632 0.01676 0.00288
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.00891 0.03106 0.15366 0.27965 0.30394 0.14667 0.02215 0.02964 0.02433 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.02557 0.07127 0.15721 0.17857 0.18355 0.20071 0.12110 0.04442 0.01760 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00337 0.00673 0.04403 0.17785 0.30641 0.25463 0.10502 0.07680 0.01887 0.00483 0.00146
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 0.01170 0.02282 0.17637 0.28225 0.23351 0.15648 0.07409 0.02159 0.01211 0.00363
0.00544 0.00000 0.00000 0.00000 0.00000 0.00000

##Number and vector of years of MRIP age compositions for general recreational fleet (single age comp for 2004-2005 from Charterboat weighted by ntrips)

#1

#2005

##Average sample sizes of pooled age compositions (first row observed Ntrips, second row Nfish; weighted; single comp for 2004-2005 Charterboat weighted by ntrips)

#28.0

#70.0

##Age composition samples (year,age) from MRIP general recreational fleet (pooled composition 2004-05 weighted by ntrips)

#0.00992 0.02493 0.16931 0.25418 0.27892 0.20430 0.03170 0.02674

#MRIP general recreational discards (pooled over mode)

#Starting and ending years of discards time series, respectively

1988

2014

#Observed general recreational discards (1000s fish) and assumed CVs (pooled over modes)

103.996 271.781 88.104 315.380 126.244 82.535 79.780 106.406 129.963 105.722 39.426 63.974 74.314
83.072 119.730 161.696 188.878 183.398 160.854 273.363 187.957 257.785 182.256 76.327 100.065
162.080 172.989

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05

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

###-- BAM DATA SECTION: parameter section

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

#

###################Parameter values and initial guesses##########################################################################

####################################################################

###prior PDF (1=none, 2=lognormal, 3=normal, 4=beta)          #

####################################################################

##initial # lower # upper #       #  prior  # prior   # prior #

## guess  # bound # bound # phase #  mean   # var/-CV #  PDF  #

##--------#-------#-------#-------#---------#---------#-------#

###### Growth curve for the population ########################## Biological input #############################################

 453.2     200.0   600.0     -4    453.2      -0.051      3  # VonBert Linf (units in mm FL)

   0.34      0.20    0.60    -4      0.34     -0.353      3  # VonBert K (units in mm FL)

  -0.98     -4.00   -0.001   -4     -0.98     -0.673      3  # VonBert t0 (units in mm FL)

   0.107    0.02     0.20    -3      0.107    -0.25       1  # stdev of length at age (sigma=8.605)


###### Growth curve for landings #################################

 474.4        200.0   700.0     4     474.4     -0.082      3   # VonBert Linf Landings (units in mm FL)

 0.21        0.05     0.60    5     0.21      -0.714      3   # VonBert K Landings(units in mm FL)

-3.73         -7.00     0.00    4    -3.73      -0.917      3   # VonBert t0 Landings(units in mm FL)

 0.096      0.02     0.20    3     0.096     -0.25       3   # stdev of length at age Landings  (sigma=8.324)


######Growth curve for SERFS FI trap survey ####################

 414.6       200.0    700.0     4     414.6     -0.062      3   # VonBert Linf Index (units in mm FL)

 0.39       0.001    1.60    5     0.39      -0.41       3   # VonBert K Index(units in mm FL)

-0.91        -10.0      0.00    4    -0.91      -0.5        3   # VonBert t0 Index(units in mm FL) (CW: -0.725)
```

```
 0.110      0.02     0.20    3     0.11      -0.25       3   # stdev of length at age Index  (sigma=8.99)

#######Natural mortality#####################################

0.41       0.02    0.6      -3     0.41       0.02       1   # constant M (used only to compute MSST=(1-M)SSBmsy)

###### SR parameters ##########################################

 0.99       0.21    0.999    -3     0.99       0.5        1   # SR steepness parameter (0.99 to give 'avg' recruitment)

#0.84       0.21    0.99     3      0.84       0.021      4  # SR steepness parameter (0.84 mode of beta distribution;
Shertzer and Conn 2012)

15.0       10.0    17.0      1     15.0       -0.25      1   # SR log_R0 parameter

 0.0       -1.0     1.0     -3     0.0        -0.5       1   # SR recruitment autocorrelation (lag 1)

 0.6        0.2     1.2      4     0.6        -0.25      3   # s.d. of recruitment in log space

###### Selectivity parameters ##################################

#Commercial handline (logistic)

3.0       1.1      9.0      2     3.0       -0.25       3    #BLOCK 1:  comm handline age at 50% selectivity

2.5       0.1     10.0      2     2.5       -0.15       3    # comm handline slope of ascending limb

#4.0       1.1      9.0      2     4.0       -0.25       3    #BLOCK 2:  comm handline age at 50% selectivity

#2.0       0.1     10.0      2     2.0       -0.25       3    # comm longline slope of ascending limb

#5.0       1.1      9.0      2     5.0       -0.25       3    #BLOCK 3: comm handline age at 50% selectivity

#2.0       0.1     10.0      2     2.0       -0.25       3    # comm handline slope of ascending limb

###### SERFS chevron trap (logistic) ##########################

4.0       1.1      9.0      2     4.0       -0.25       3    # SERFS chevron trap age at 50% selectivity

3.0       0.1     10.0      2     3.0       -0.15       3    # SERFS chevron trap slope of ascending limb

#Headboat (double logistic with option for 2 separate logistic exponentials)
```

```
#Headboat age at peak selectivity = 3 yrs (1990, 2003, 2005, 2006, 2010, 2012, 2013, 2014)
2.0   0.5   9.0   2   2.0   -0.25   3   # BLOCK 1: HB age at 50% selectivity (ascending)
2.0   0.1   10.0  3   2.0   -0.15   3   # HB slope of ascending limb
3.0   1.0   14.0  2   3.0   -0.25   3   # HB age at 50% selectivity (descending)
1.0   0.1   10.0  5   1.0   -0.15   3   # HB slope of descending limb (rate of descent)

#Headboat age at peak selectivity = 4 yrs ####################
#(1991, 2007, 2008, 2009, 2011)
2.5   0.5   9.0   -2  2.5   -0.25   1   #BLOCK 1: HB age at 50% selectivity
2.0   0.1   10.0  -2  2.0   -0.25   1   # HB slope of ascending limb
4.0   1.0   14.0  -2  4.0   -0.25   1   # HB age at peak selex = 1
2.0   0.1   10.0  -3  2.0   -0.25   1   # HB descending limb (rate of descent)

#4.0   1.1   9.0   2   4.0   -0.25   3   #BLOCK 2: HB age at 50% selectivity (logistic)
#2.0   0.1   10.0  2   2.0   -0.25   3   # HB slope of ascending limb

#5.0   1.1   9.0   2   5.0   -0.25   3   #BLOCK 3: HB age at 50% selectivity (logistic)
#2.0   0.1   10.0  2   2.0   -0.25   3   #HB slope of ascending limb

###### Headboat discards (logistic exponential) ###############
1.0   0.1   9.0   2   1.0   -0.25   3   # HB discards age at 50% selectivity
2.0   0.1   10.0  3   2.0   -0.15   3   # HB discards slope of ascending limb
1.0   0.5   8.0   -2  1.0   -0.25   3   # HB discards  age at peak selex = 1
1.0   0.1   10.0  5   1.0   -0.15   3   # HB discards  descending limb (rate of descent)

###### MRIP General recreational (double logistic) ############
3.0   0.5   9.0   2   3.0   -0.25   3   # GR age at 50% selectivity (ascending)
3.0   0.1   10.0  3   3.0   -0.15   3   # GR slope of ascending limb
3.0   1.0   14.0  2   3.0   -0.25   3   # GR age at 50% selectivity (descending)
```

```
1.0      0.1     10.0      5     1.0      -0.15      3     # GR slope of descending limb (rate of descent)


###### Index catchability parameters #########################

-8.5    -16.0     -4.0      -1    -8.5      -0.5       1      # comm handline CPUE (log q)

-8.5    -16.0     -4.0       1    -8.5      -0.5       1      # SERFS chevron trap CPUE (log q)

#-8.5    -16.0     -4.0      -1    -8.5      -0.5       1     # SERFS video CPUE (log q)

-13.5   -16.0     -8.0      -1   -13.5      -0.5       1      # HB CPUE (log q)

-13.5   -16.0     -8.0      -1   -13.5      -0.5       1      # GR CPUE (log q)


###### Fishing mortality parameters ###########################

 0.05      0.001   2.2       1     0.05     -0.5       3      # F used to initialize popn, distributed among fleets in proportion to
their early Fs

-2.0       -8.0    1.0       1    -2.0      -0.5       1      # comm handlines log mean F

-4.0      -10.0    1.0       1    -4.0      -0.5       1      # HB log mean F

-3.0      -10.0    1.0       1    -3.0      -0.5       1      # GR log mean F

-4.0      -11.0    1.0       1    -4.0      -0.5       1      # HB discards log mean F

-4.0      -10.0    1.0       1    -4.0      -0.5       1      # GR discards log mean F


##### Dev vectors
#################################################################################################

#########################

# lower # upper #       #

# bound # bound # phase #

#-------#-------#-------#

-10.0     5.0      2     # comm handline F devs

-10.0     5.0      2     # HB F devs

-10.0     5.0      2     # GR F devs

-10.0     5.0      2     # HB discard F devs

-10.0     5.0      2     # GR discard F devs

 -5.0     5.0      2     # recruitment devs
```

-5.0 5.0 3 # Nage devs

#Commercial handline F dev initial guesses

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0

#HB F dev initial guesses

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0

#GR F dev initial guesses

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0

#HB discard F dev initial guesses

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0

#GR discard F dev initial guesses

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0

#Rec devs

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0

#Initial N age devs, all ages but the first one

0.0 0.0 0.0 0.0 0.0 0.0 0.0

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

##-- BAM DATA SECTION: likelihood weights section

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

##################Likelihood Component Weighting##

1.0 #landings

1.0 #discards

1.0 #cH index

3.6 #0.60 #mcvt index (6X iterative re-weighting)

#1.0 #video index

1.0 #HB index

1.0 #MRIP GR index

0.06#0.13 #cH len comps

0.072 #SERFS trap (mcvt) len comps

0.04#0.075 #HB len comps

0.40 #HB discard len comps

0.20 #MRIP GR len comps

0.15 #cH age comps

0.092 #SERFS trap (mcvt) age comps

0.07 #HB age comps

#1.0 #MRIP age comp (CB mode pooled over 2004-05, weighted by Ntrips)

1.0 #log N.age.dev residuals (initial abundance)

1.0 #S-R residuals

0.0 #constraint on early recruitment deviations

0.0 #constraint on ending recruitment deviations

0.0 #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization) fULL F summed over fisheries

0.0 #weight on tuning F (penalty not applied in final phase of optimization)

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

##-- BAM DATA SECTION: miscellaneous stuff section

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

#

#Length-weight (FL-whole wgt) coefficients a and b, W=aL^b, (W in kg, FL in mm)--sexes combined

2.75E-08

2.97

#Fork length-fecundity (FL-batch fecundity) coefficients a and b, fec=a+b*FL (units=mm)

-1.776483E06

8703.7

#Batches per yr

0.1 2.3 3.45 4.6 6.4 7.6 7.6 7.6 #age-dependent batches per year

#5.11 5.1 5.1 5.1 5.1 5.1 5.1 5.1 #constant batches per year

#Used to scale annual fecundity

1.0e6

#Time-invariant vector of % maturity-at-age for females (ages 1-8+)

0.793 0.953 0.991 0.998 1.000 1.000 1.000 1.000

#Time-invariant vector of proportion female (ages 1-8+)

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

#Time of year (as fraction) for spawning

0.497

#Age-dependent natural mortality at age (ages 1-8+) -- scaled Charnov (2012) to Then et al. (2015) point estimate M=0.41

0.887 0.660 0.552 0.491 0.455 0.431 0.416 0.405

#0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2

#Max observed age

15

#Discard mortality

0.125 #HB (for combined HB discards)

0.125 #GR (for MRIP GR discards)

#Spawner-recruit parameters

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

1

#Switch for rate increase in q: Integer value (choose estimation phase, negative value turns it off)

-1

#Annual positive rate of increase on all fishery dependent q's due to technology creep

0.02

#DD q switch: Integer value (choose estimation phase, negative value turns it off)

-1

#Density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)

0.0

#SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)

0.128

#Age to begin counting D-D q (should be age near full exploitation)

4.0

#Random walk switch:Integer value (choose estimation phase, negative value turns it off)

-3

#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence)

0.03

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

0.35

#Year for tuning F

2000

#Threshold sample sizes ntrips (>=)for length comps (set to 99999.0 if sel is fixed):

1.0 #comm cH

1.0 #SERFS chevron trap (mcvt) lengths

1.0 #HB landings lengths

1.0 #HB discards lengths

1.0 #GR lengths

#Threshold sample sizes ntrips (>=) for age comps (set to 99999.0 if sel is fixed)

1.0 #comm cH

1.0 #SERFS chevron (mcvt) trap

1.0 #HB

#1.0 #GR (MRIP)

#Ageing error matrix (columns are true age 1-8+, rows are ages as read for age comps: columns should sum to one)

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

0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0

#USE LANDINGS GROWTH CURVE FLAG USE 0.0 for the population growth rate and 1.0 for the fishery landings growth rate

1.0

999 #end of data file flag