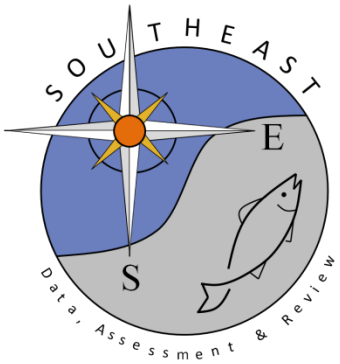
Red Snapper: Additional BAM diagnostics, analyses, and code

Sustainable Fisheries Branch, National Marine Fisheries Service
(Contact: Katie Siegfried)

SEDAR41-RW04

Submitted: 29 February 2016
Updated: 16 March 2016



Please cite this document as:

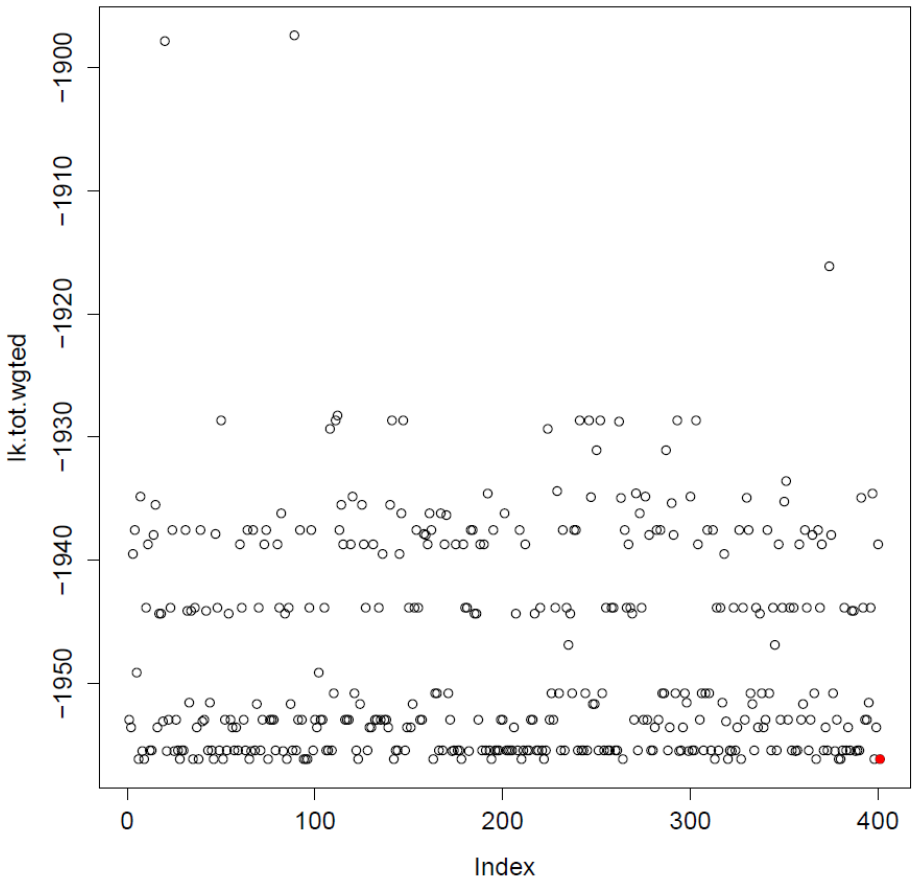
SFB-NMFS. 2016. Red Snapper: Additional BAM diagnostics, analyses, and code. SEDAR41-RW04. SEDAR, North Charleston, SC. 205 pp.

Additional BAM diagnostics, analyses, and code

This working paper is meant as a repository for all supplementary materials to the assessment. The plots are all diagnostics or results plotted in an alternative way, and the code that follows the plots is the source code and data file for the catch-age model.

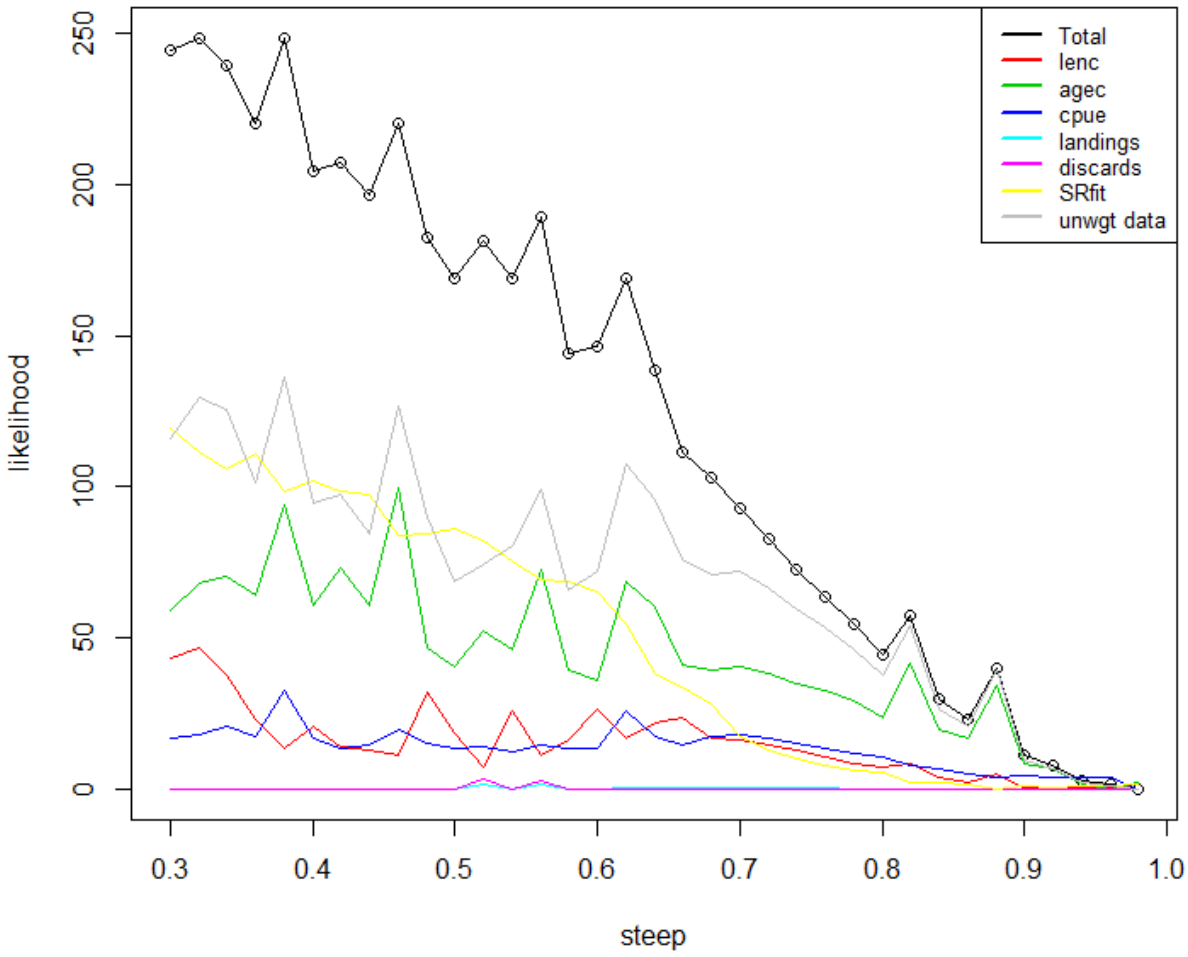
Starting Value Analysis:

This analysis was used during SEDAR 41 to determine whether the BAM was sensitive to the starting values provided for the estimated parameters. For the analysis, all starting values for estimated parameters were randomly drawn from a uniform distribution with a lower and upper bound set to 25% of the original starting values. For each bootstrap, a random uniform starting value was drawn and the BAM model was fit. The drawn value, the corresponding estimate, and the overall model likelihood were tabulated and compared at the end of 400 trials. If the model was not reaching the global minimum, the starting values were set to the values in the run that did reach the global minimum and the analysis was repeated. For Red Snapper, the global minimum was found using this analysis, and the starting values reflect the change from the original values. This plot shows the 400 bootstrap trials with the base model result in red.

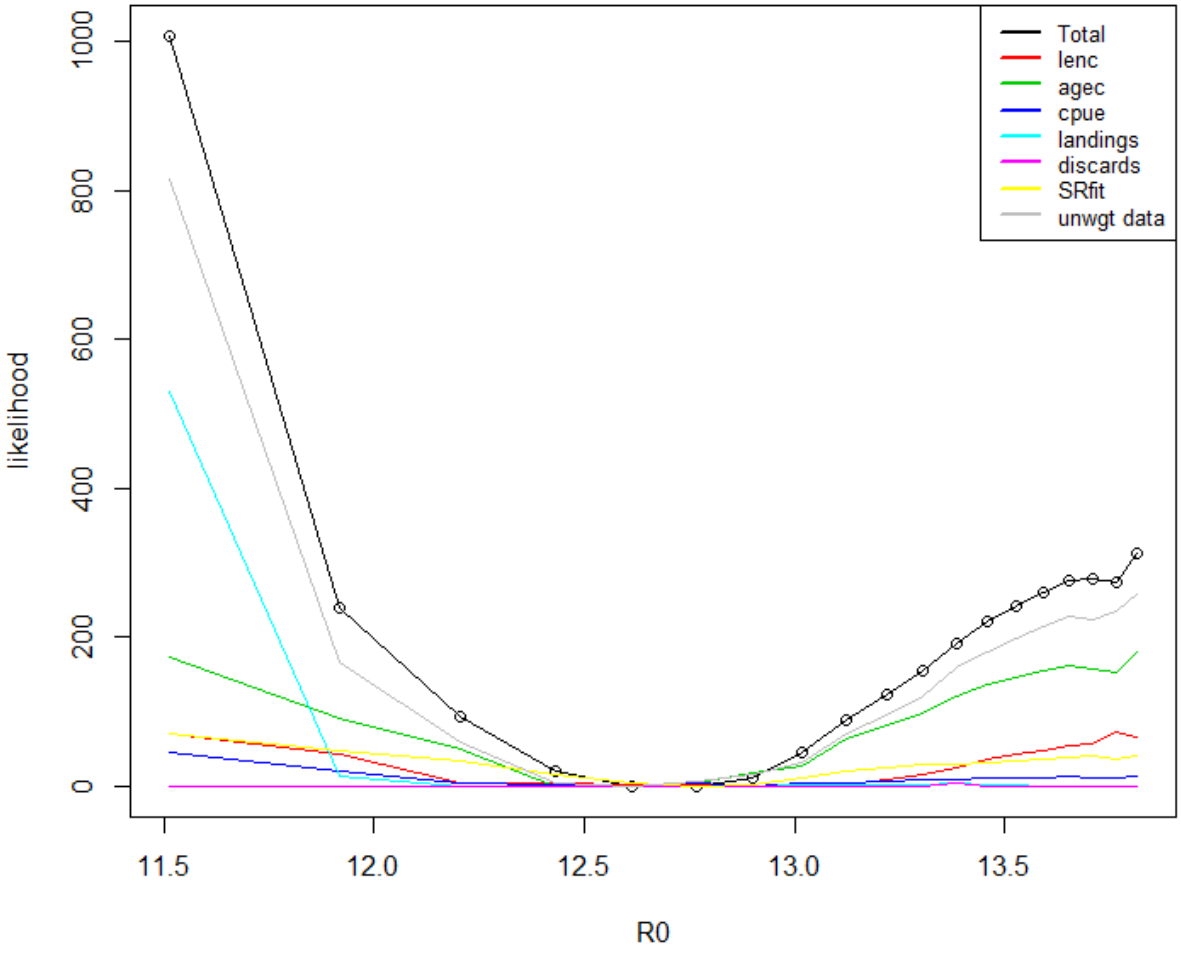


Likelihood profiles:

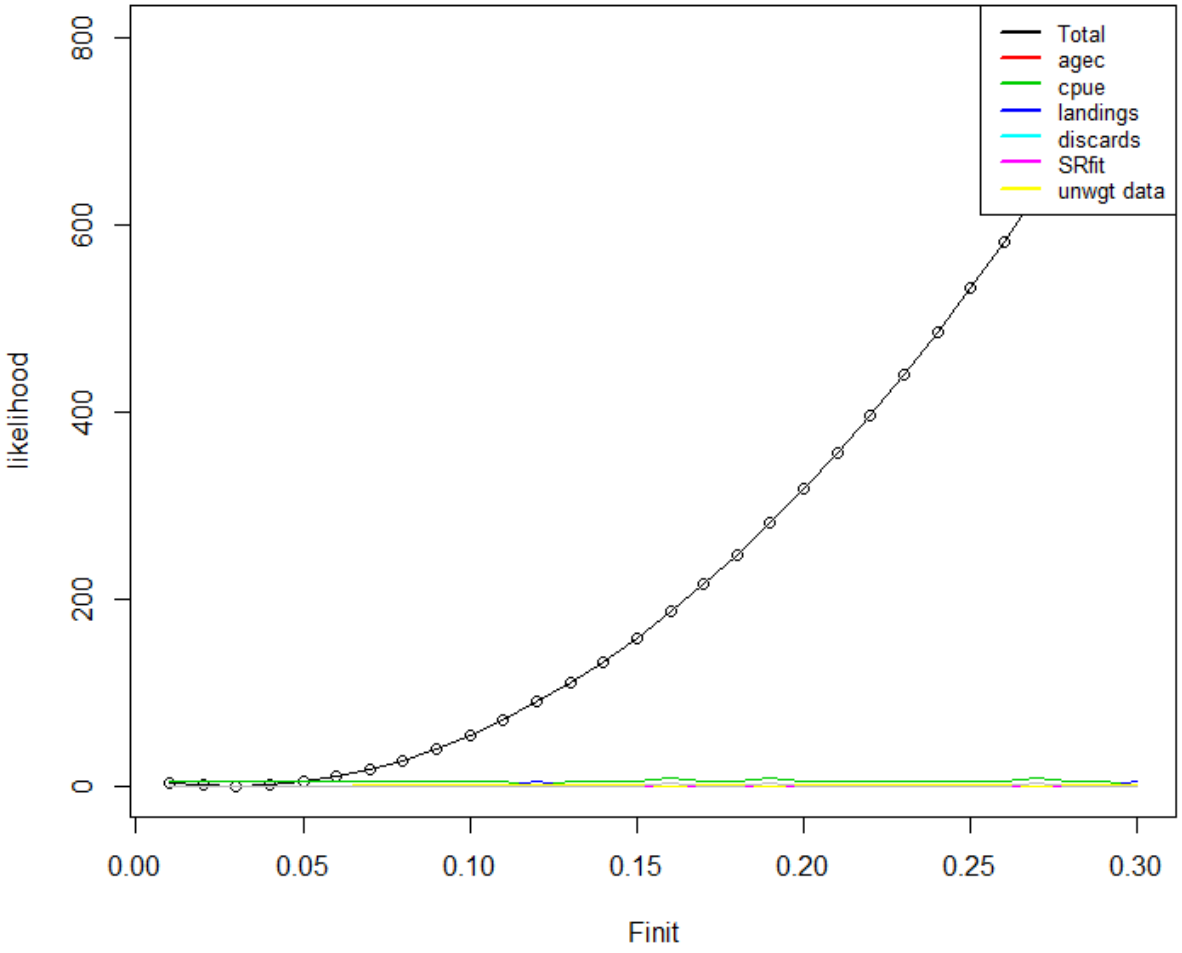
A range of steepness values were used over which to profile (h=0.3 to 0.98):



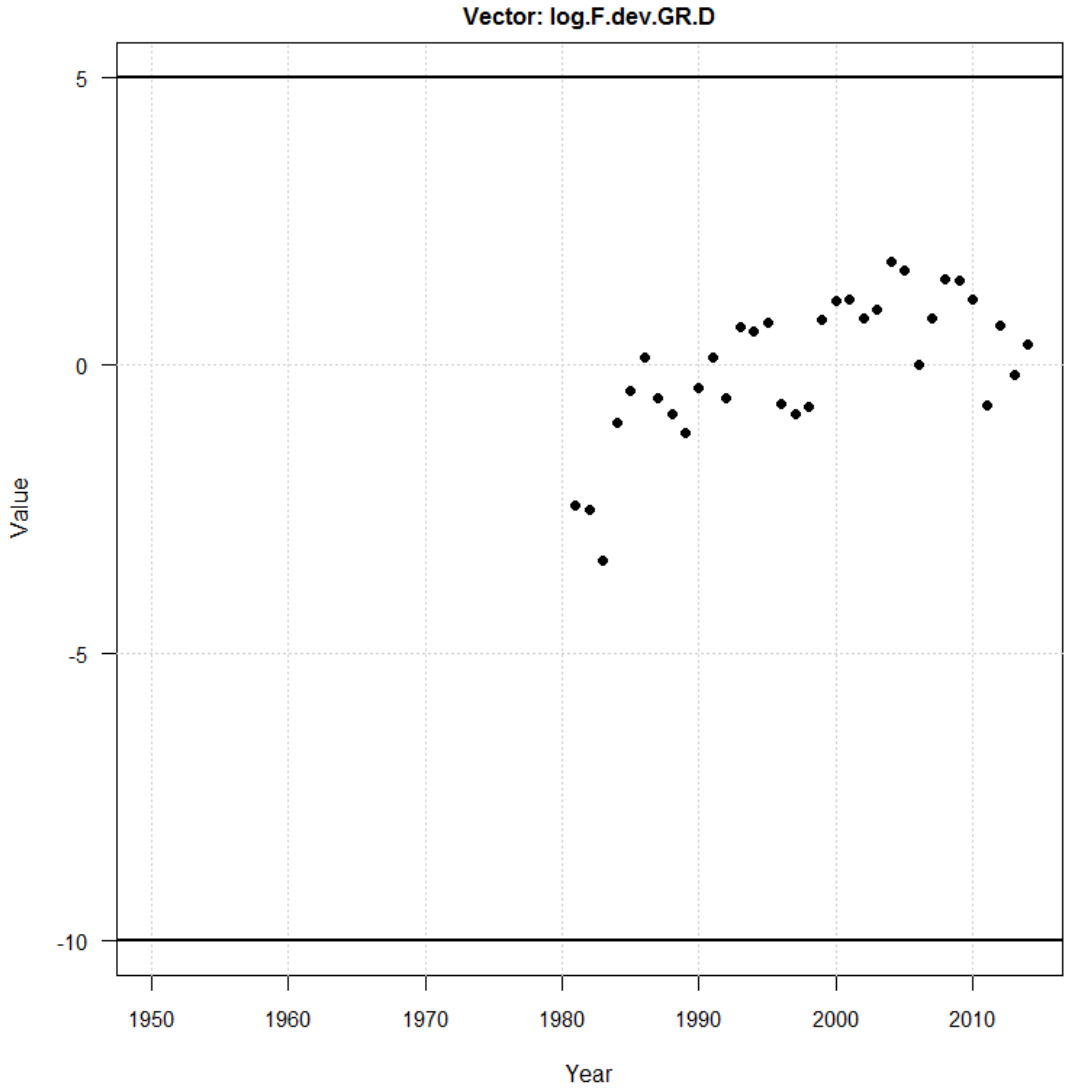
R0 was profiled using a range from 100,000 to 1,000,000 (11.5 to 14 on a log scale):

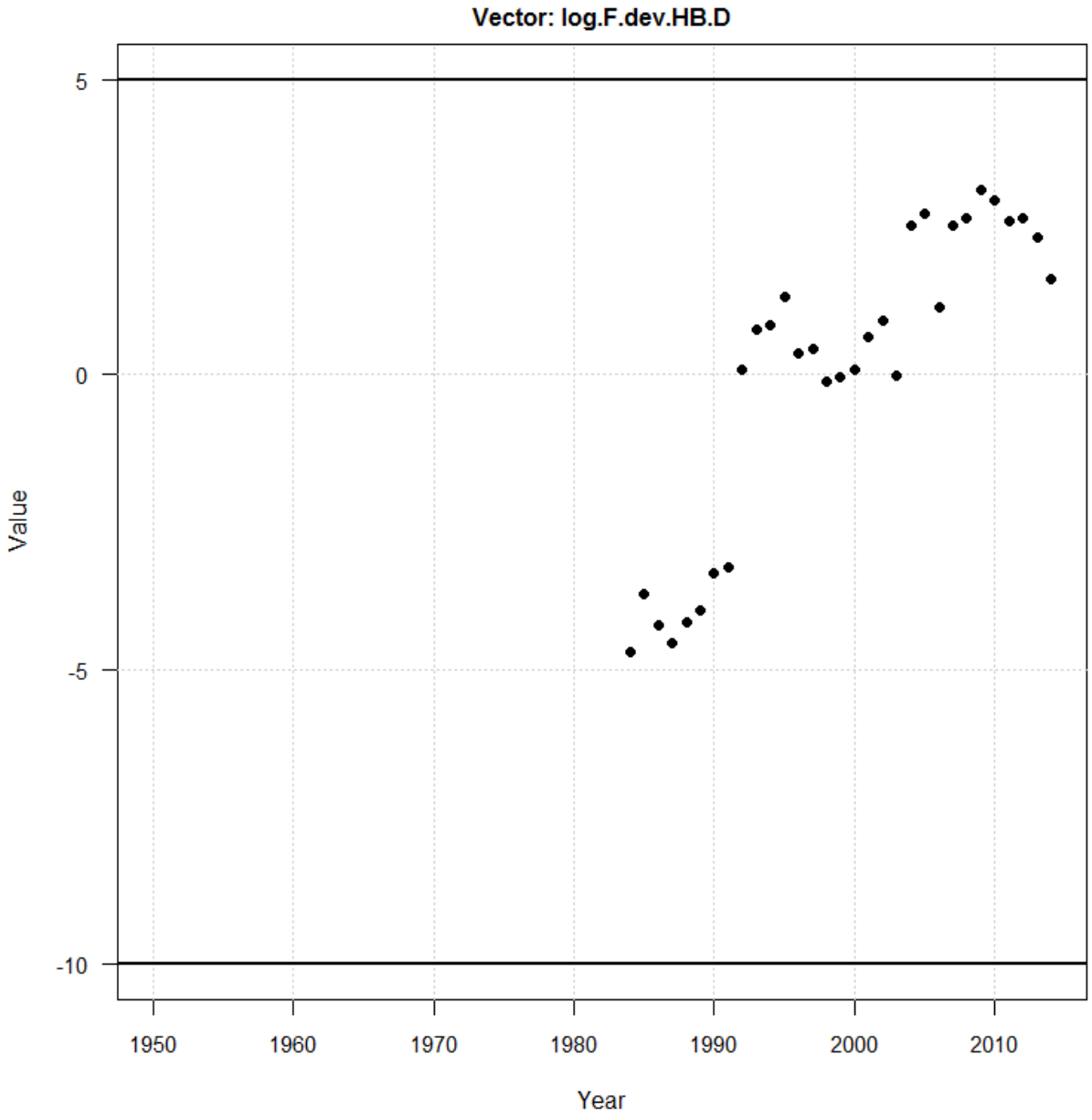


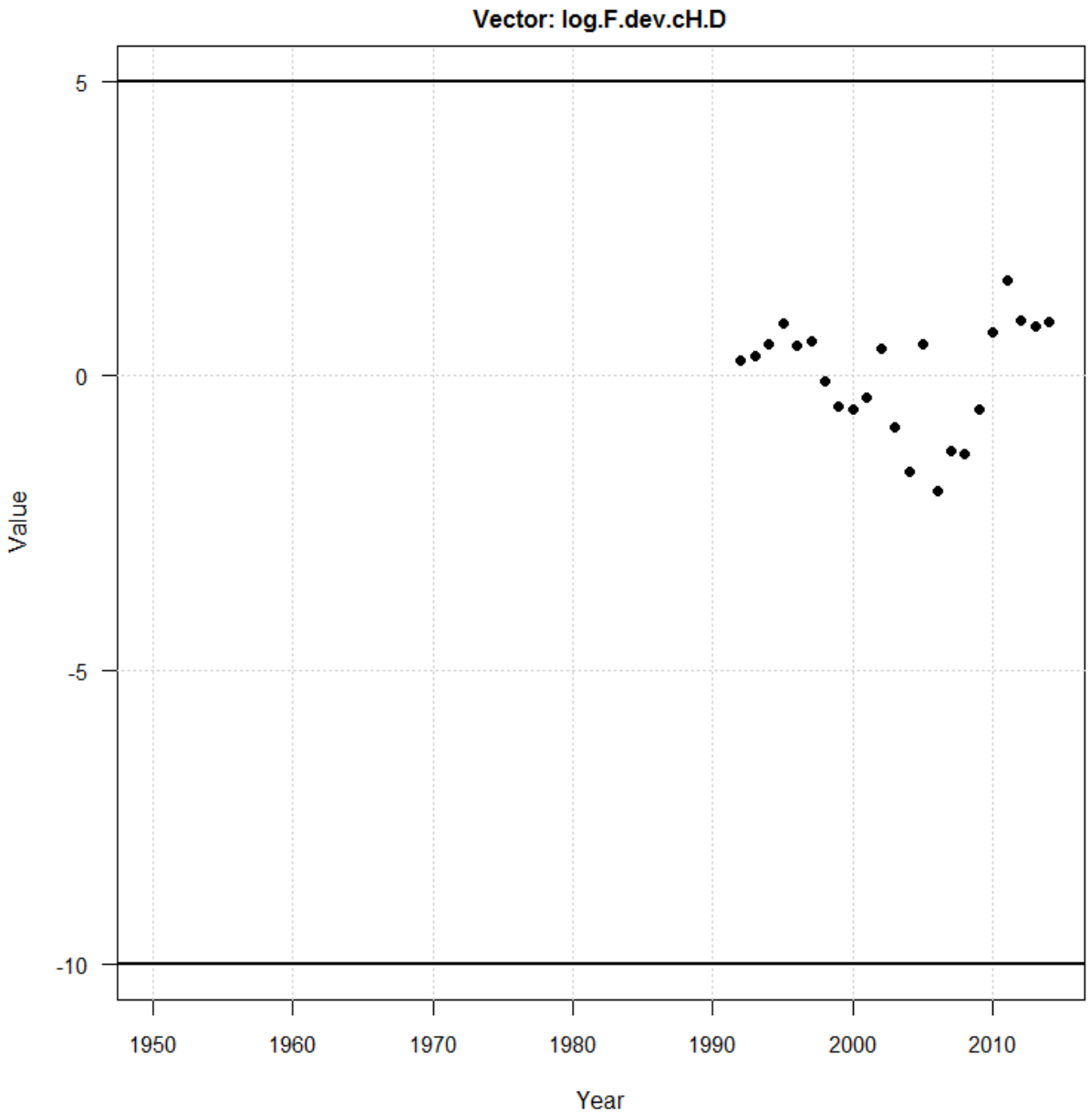
Initial F was profiled using a range from 0.01 to 0.03:

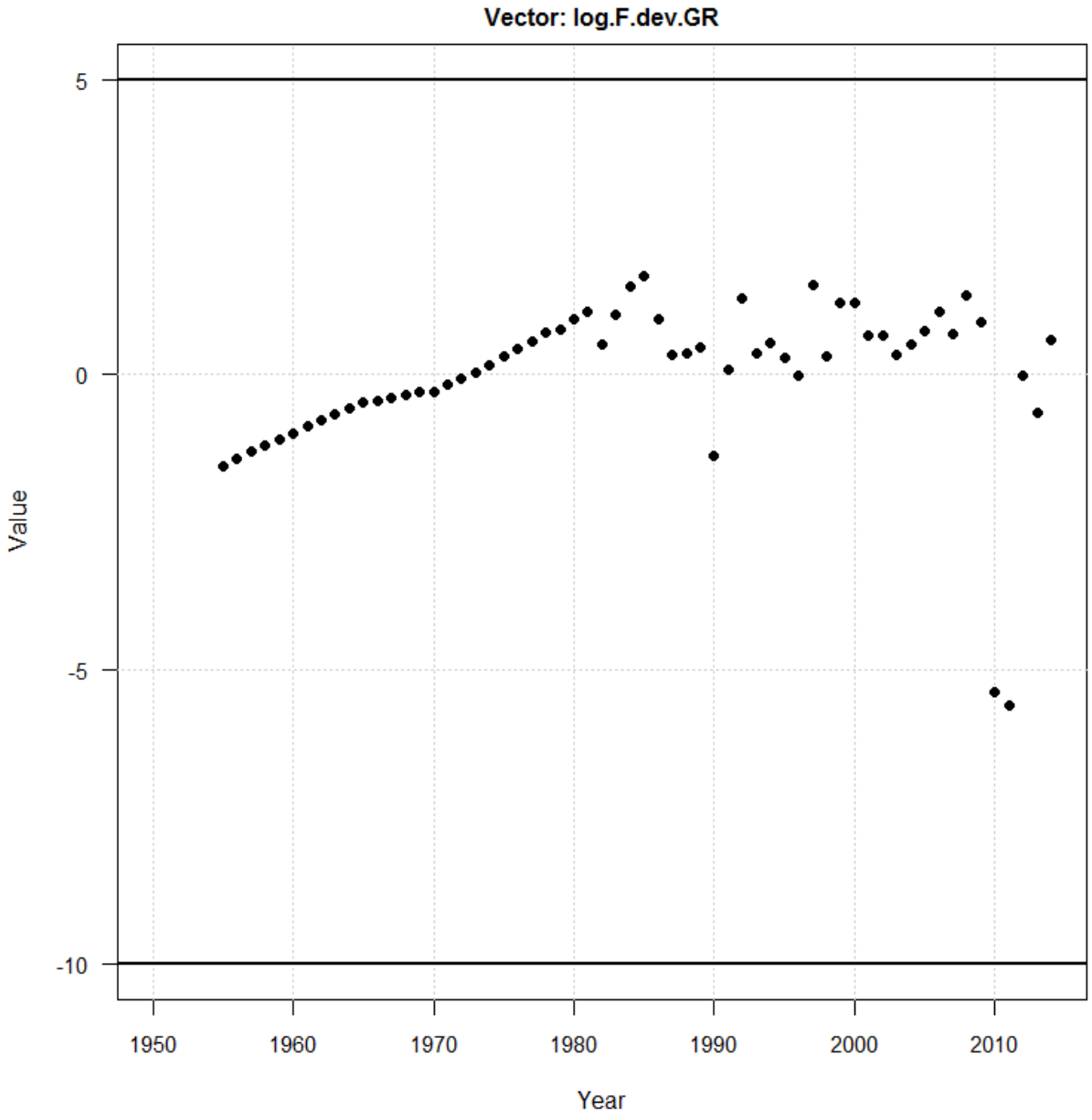


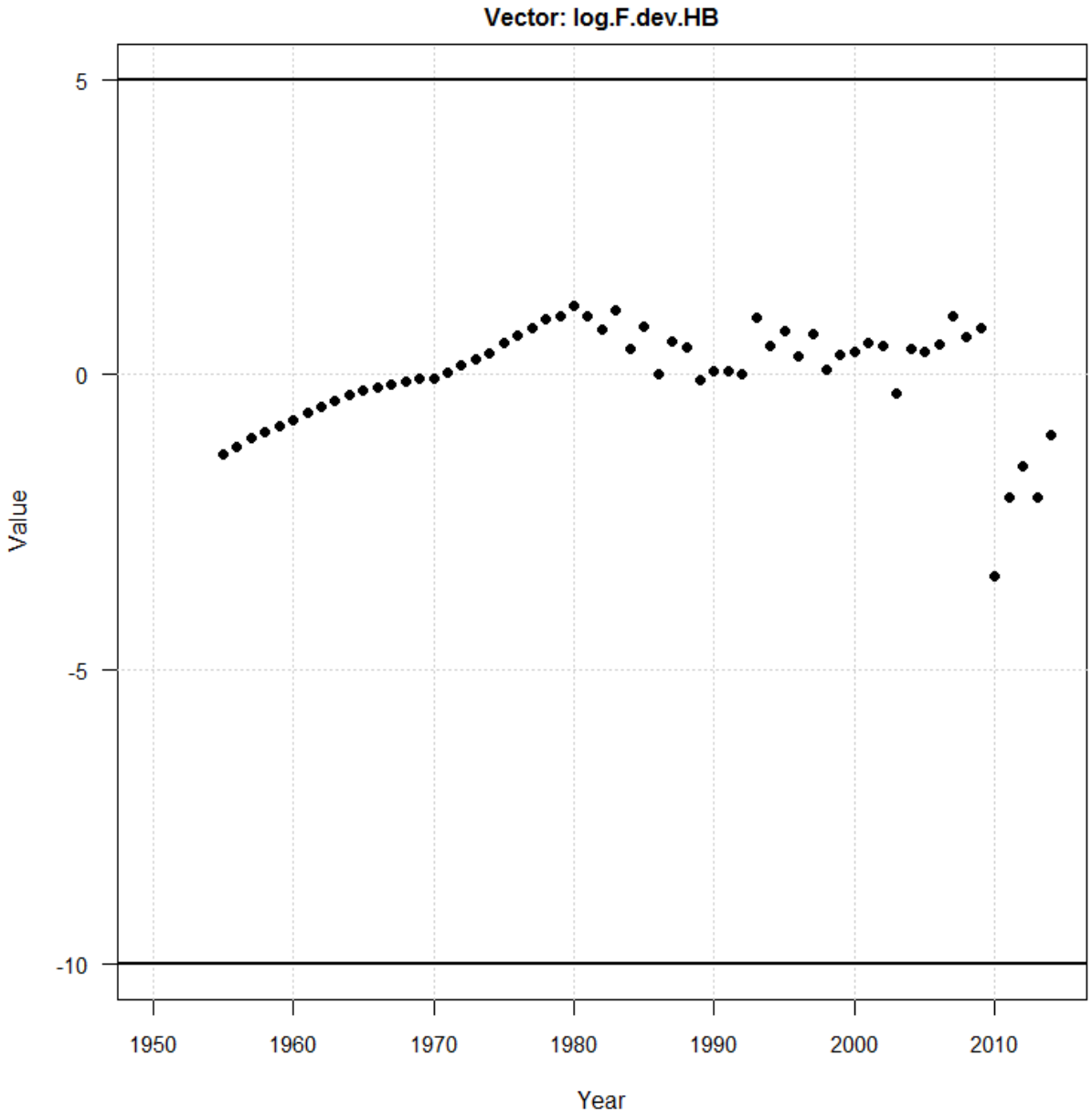
Diagnostic and supplementary plots:

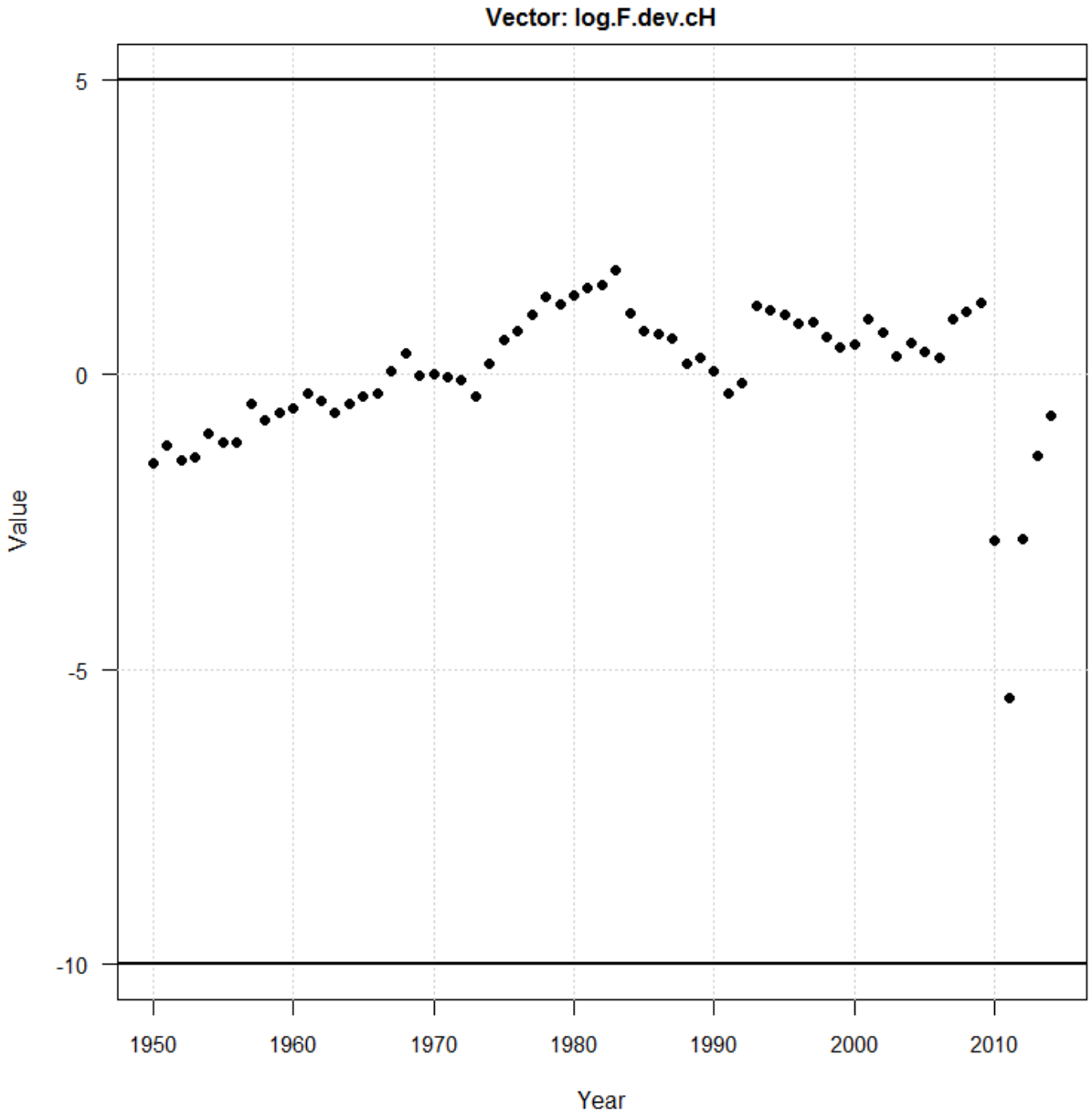


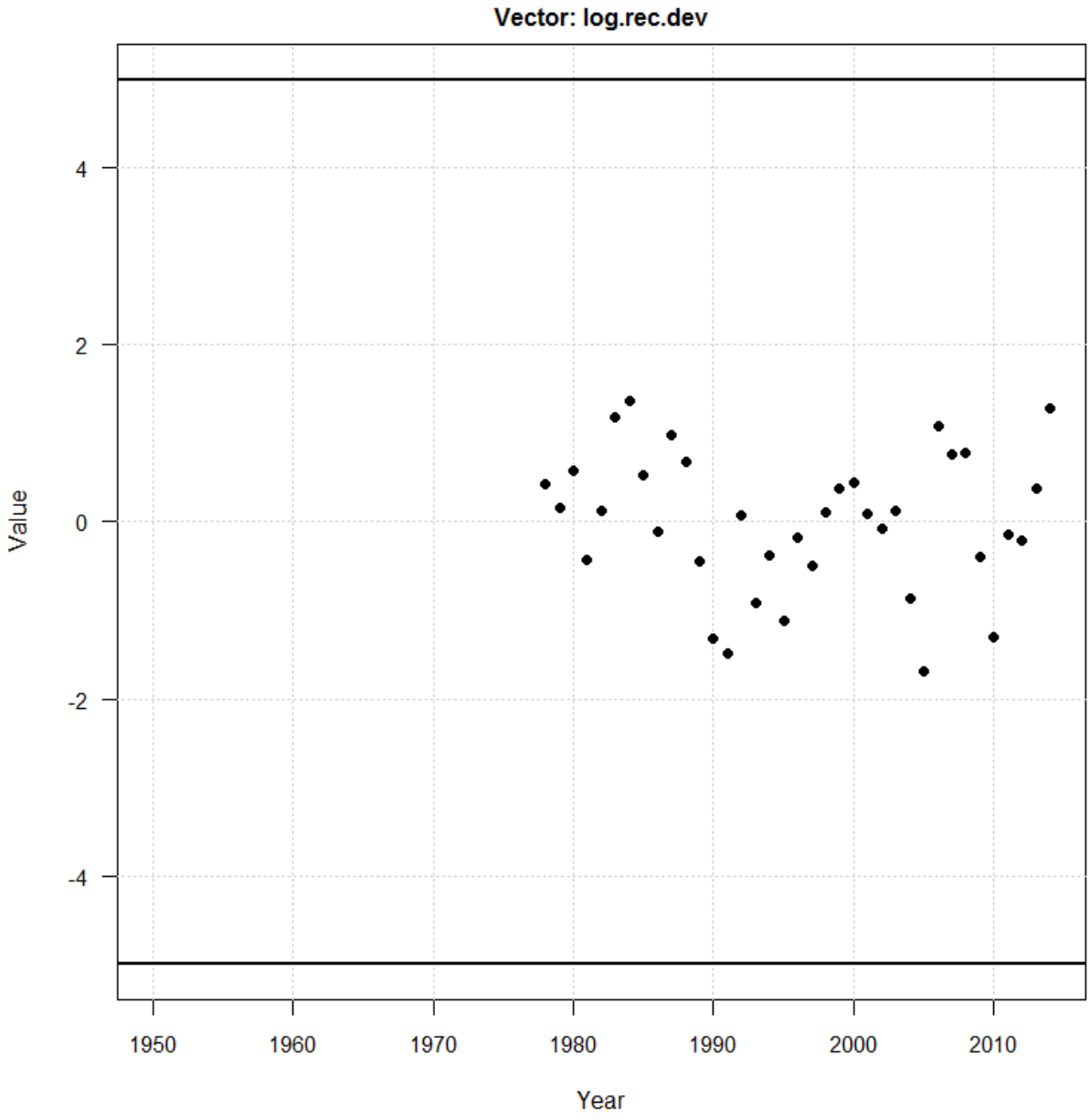

Vector: log.F.dev.HB.D
5
0
-5
-10
Value
1950
1960
1970
1980
1990
2000
2010
Year

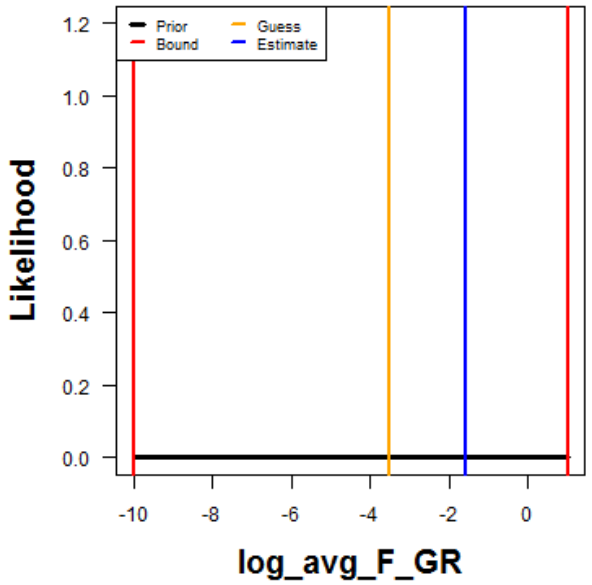

Vector: log.F.dev.cH.D
5
0
-5
-10
Value
1950
1960
1970
1980
1990
2000
2010
Year

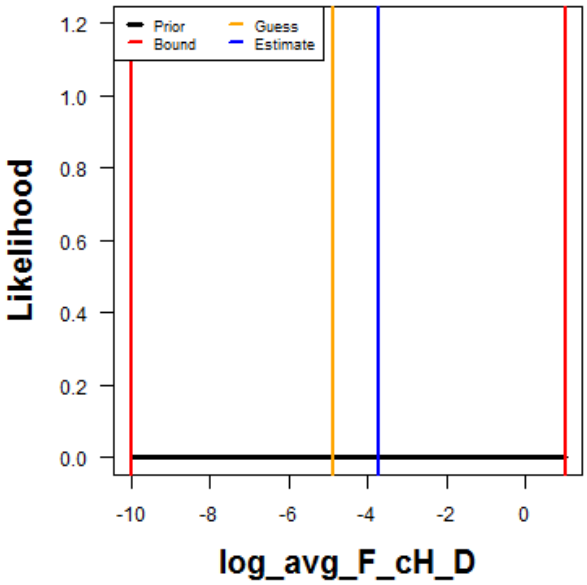

Vector: log.F.dev.GR
5
0
-5
-10
Value
1950
1960
1970
1980
1990
2000
2010
Year

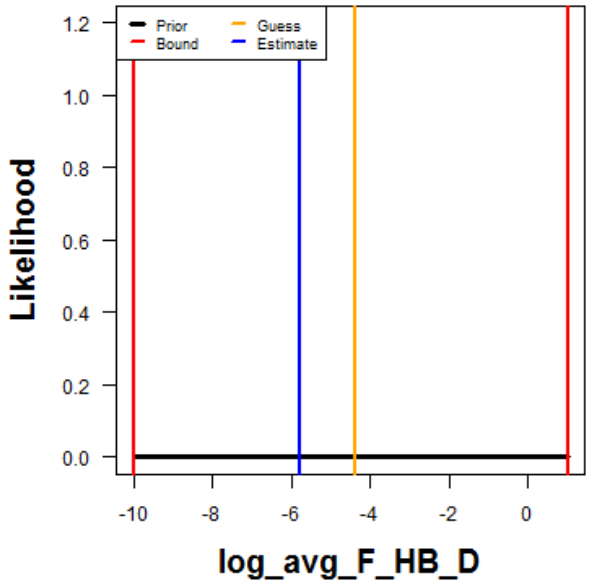

Vector: log.F.dev.HB
5
0
-5
-10
Value
1950
1960
1970
1980
1990
2000
2010
Year

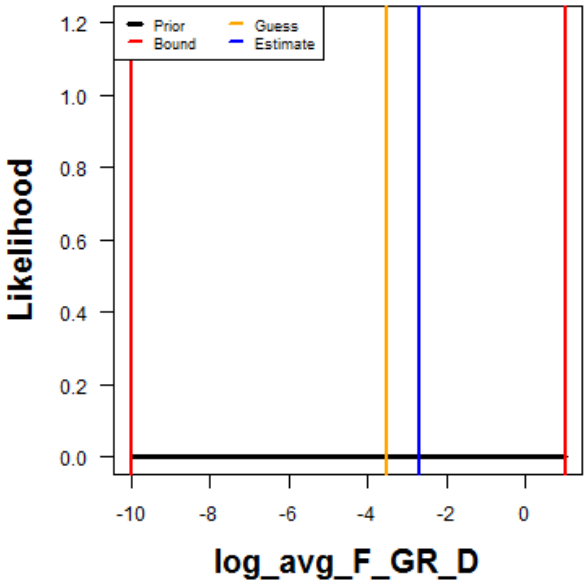

Vector: log.F.dev.cH
5
0
-5
-10
Value
1950
1960
1970
1980
1990
2000
2010
Year

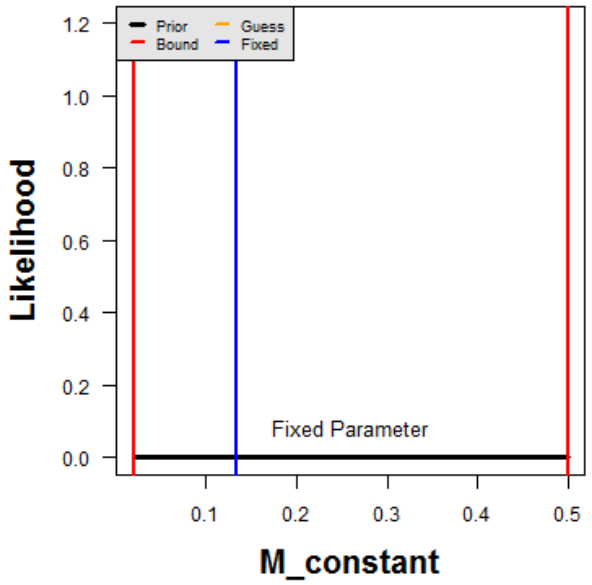

Vector: log.rec.dev
Value
4
2
0
-2
-4
1950
1960
1970
1980
1990
2000
2010
Year

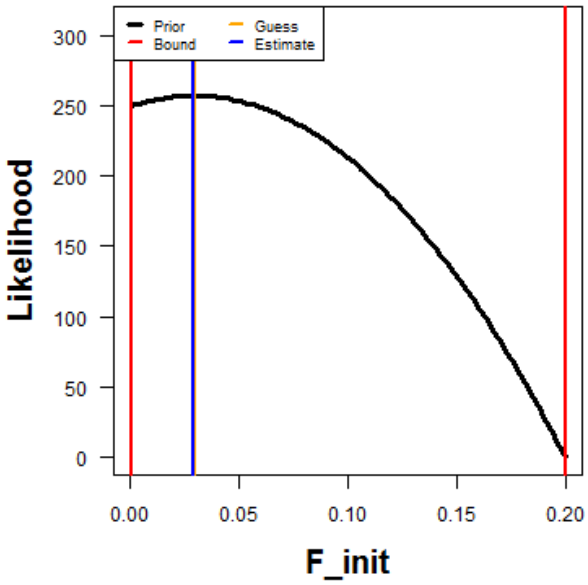

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_GR

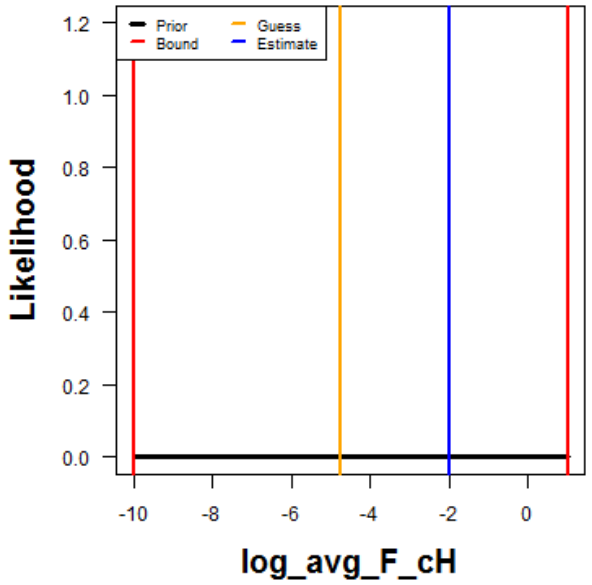

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_cH_D

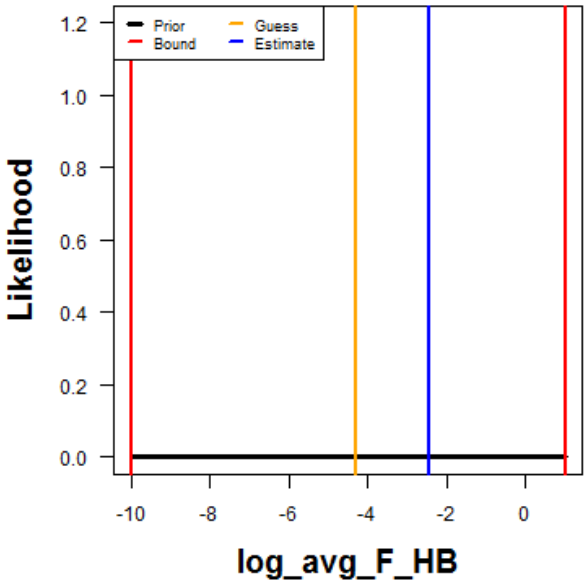

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_HB_D

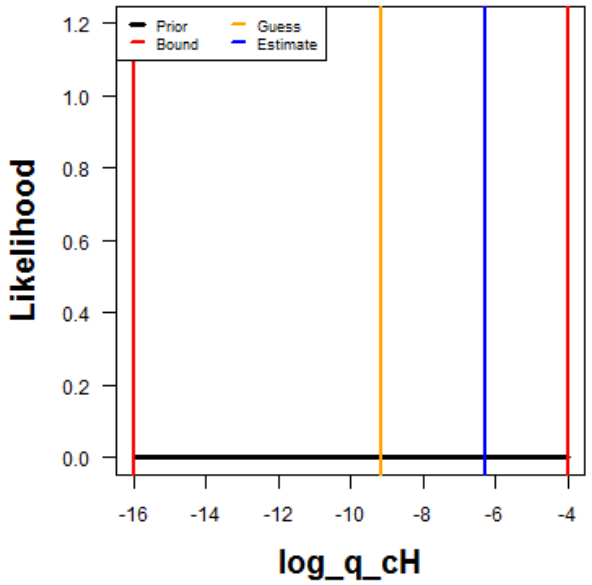

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-10
-8
-6
-4
-2
0
log_avg_F_GR_D

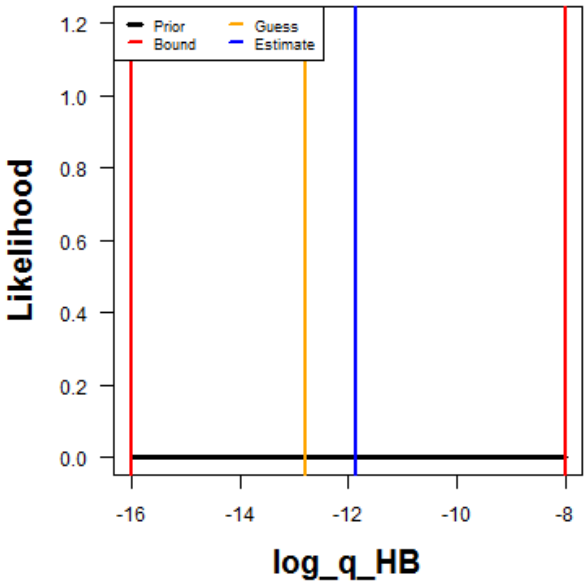

Prior
Bound
Guess
Fixed
Likelihood
Fixed Parameter
M_constant

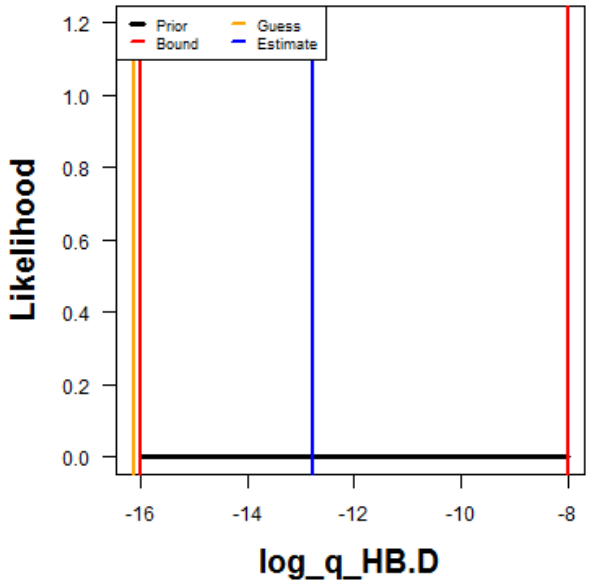

Prior
Bound
Guess
Estimate
Likelihood
F_init

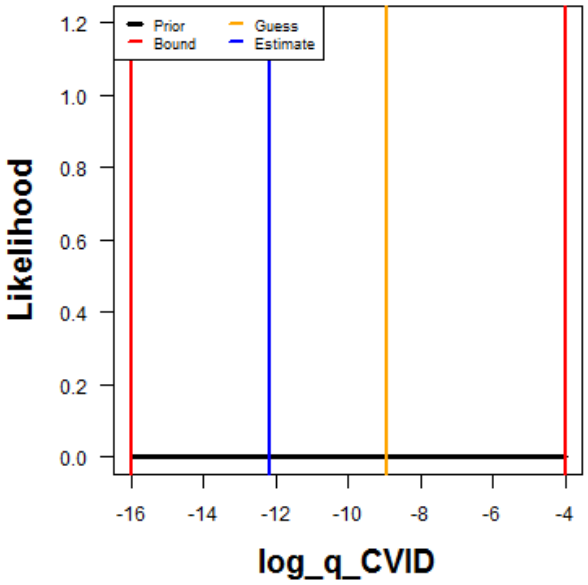

Prior
Bound
Guess
Estimate
Likelihood
log_avg_F_cH

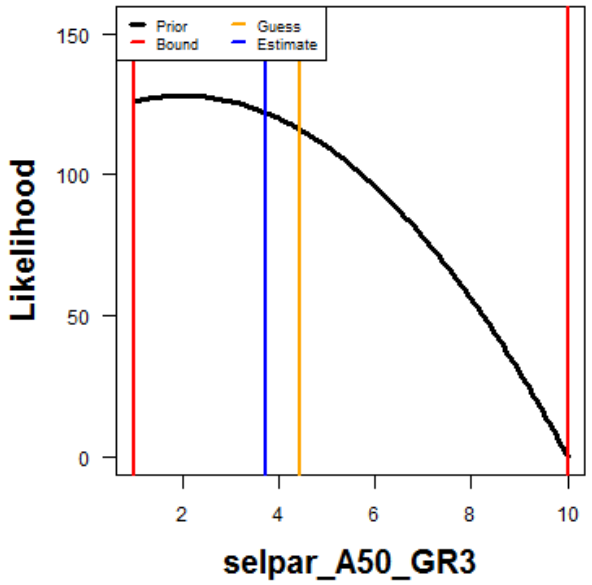

Prior
Bound
Guess
Estimate
Likelihood
log_avg_F_HB

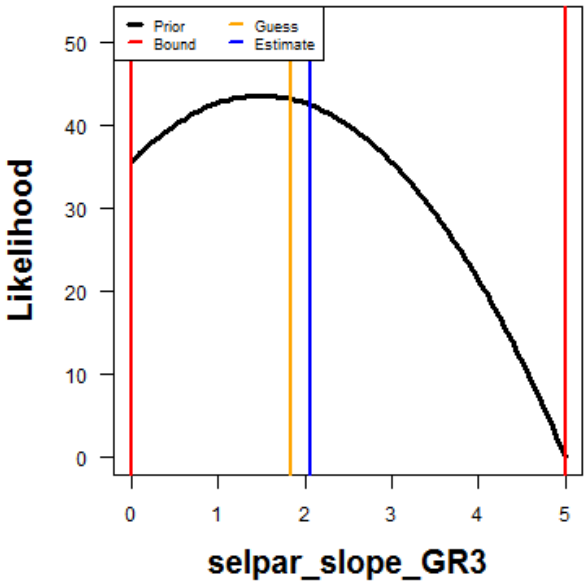

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-16
-14
-12
-10
-8
-6
-4
log_q_cH

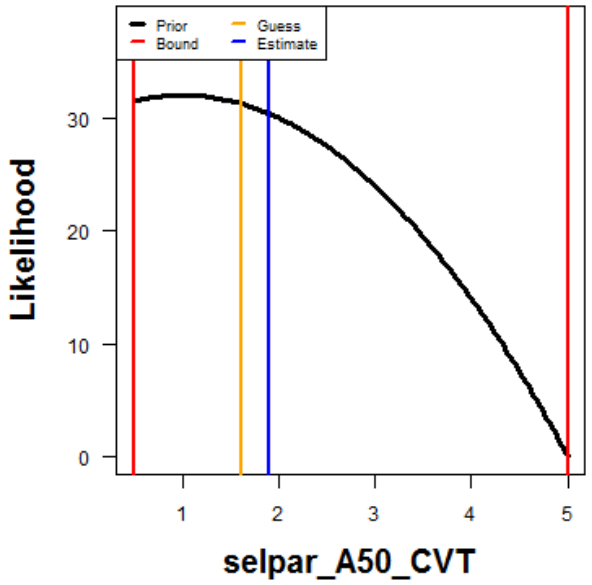

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-16
-14
-12
-10
-8
log_q_HB

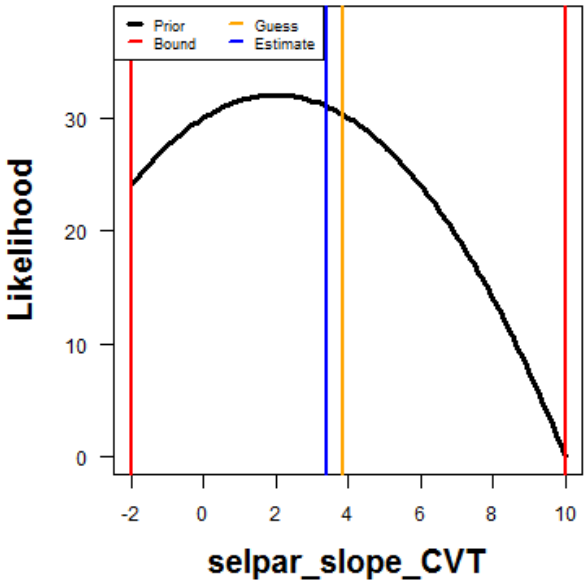

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-16
-14
-12
-10
-8
log_q_HB.D

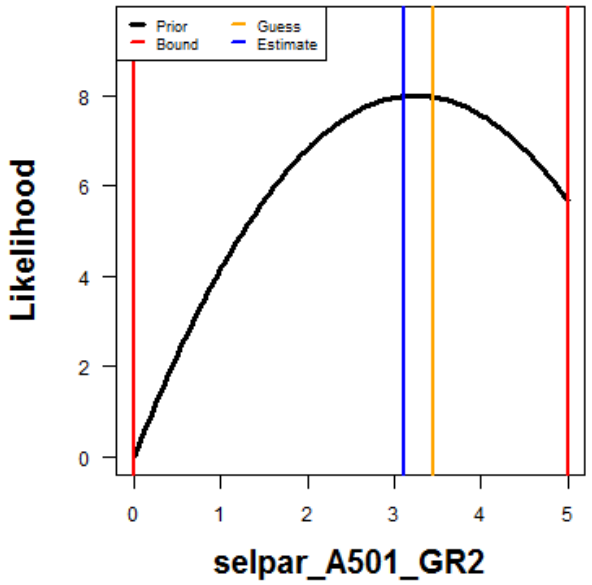

Prior
Bound
Guess
Estimate
Likelihood
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-16
-14
-12
-10
-8
-6
-4
log_q_CVID

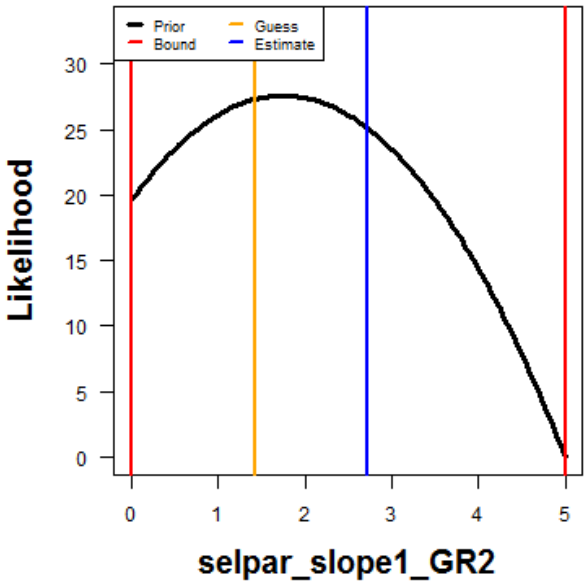

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_GR3

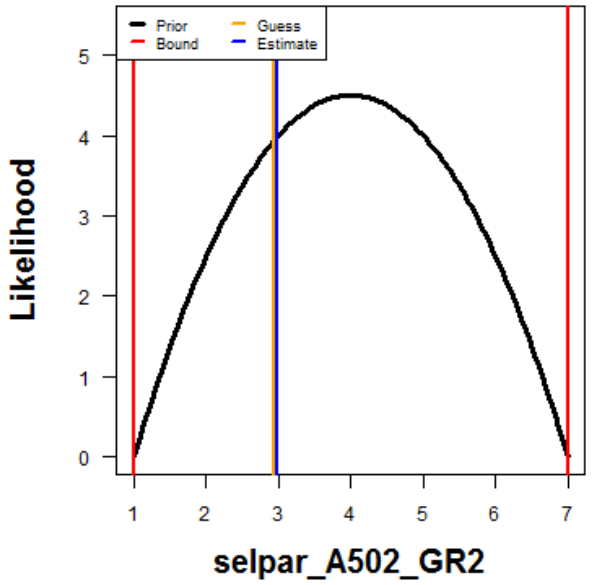

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_GR3

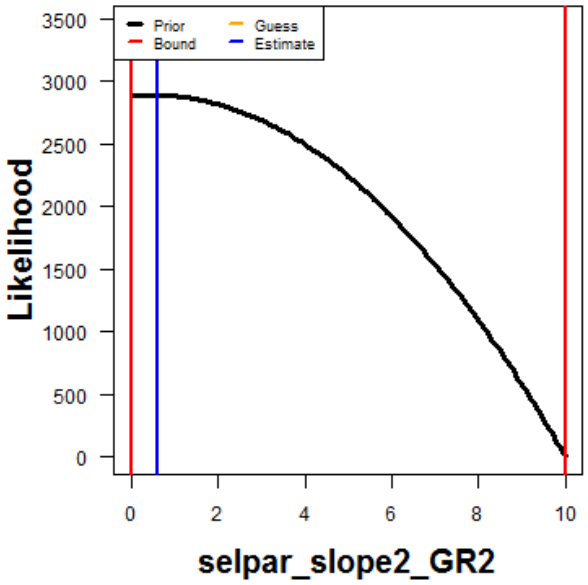

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_CVT

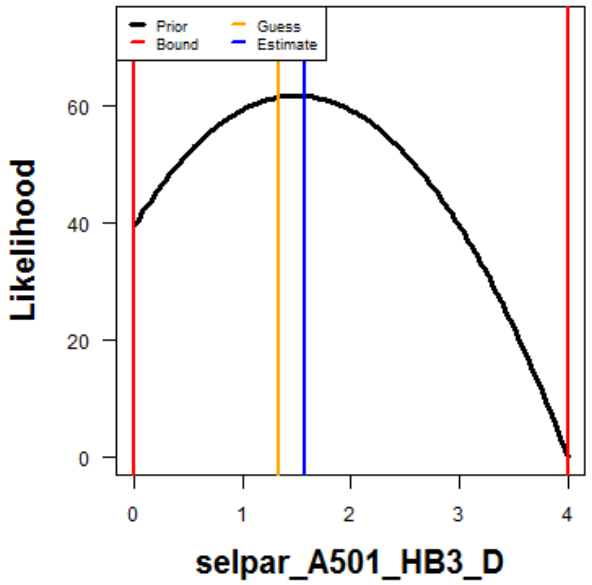

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_CVT

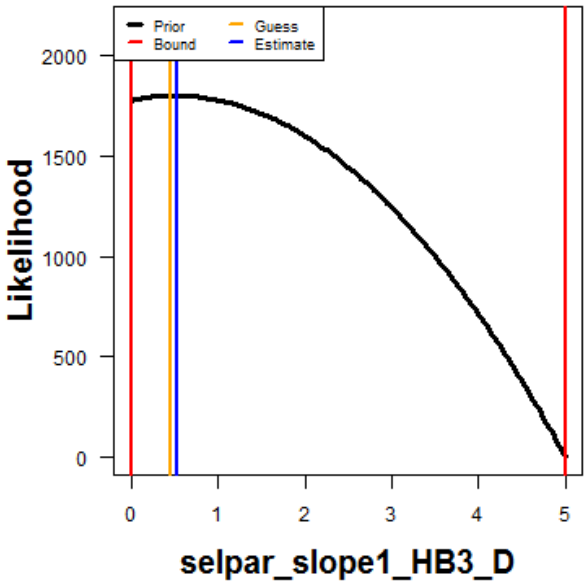

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_GR2

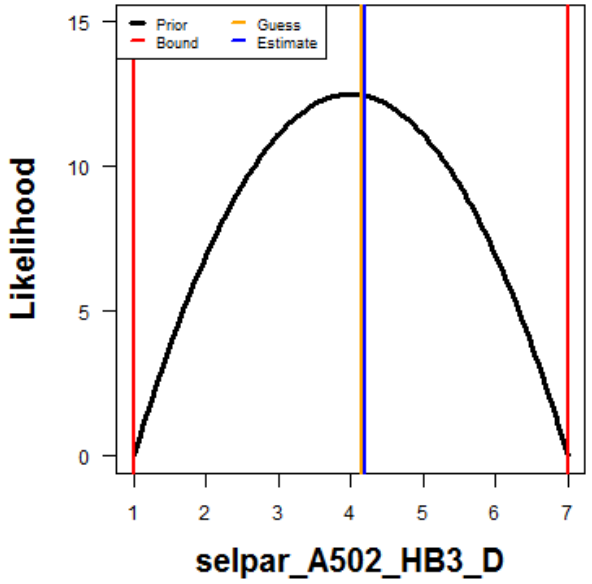

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_GR2

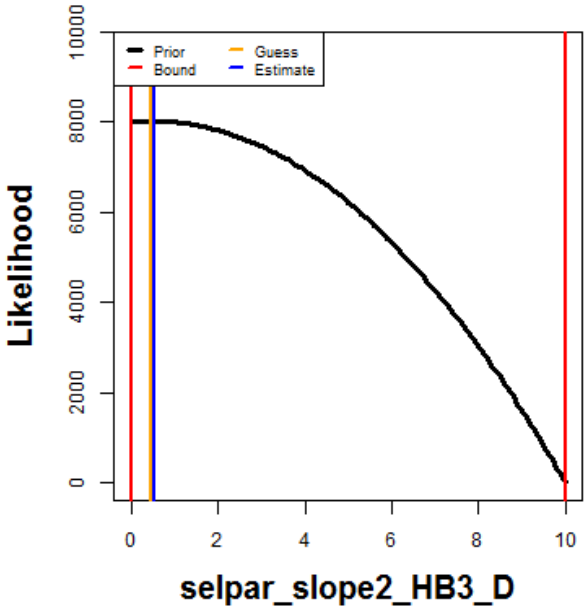

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_GR2

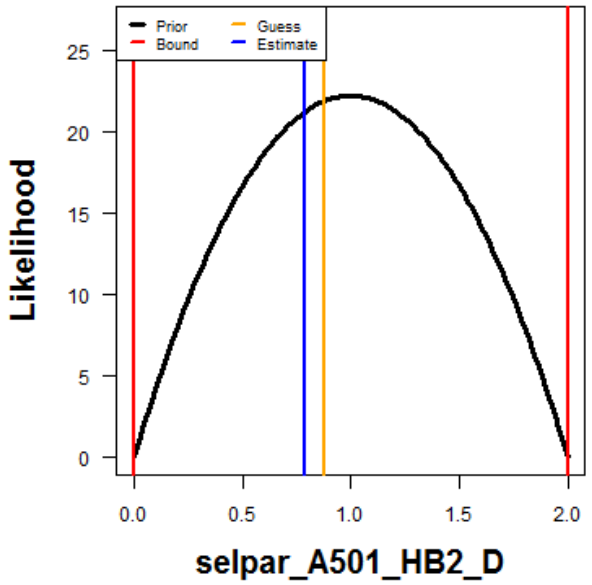

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_GR2

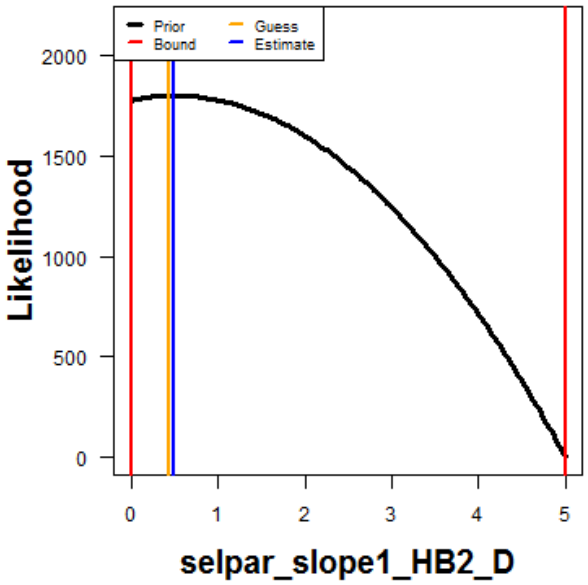

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_HB3_D

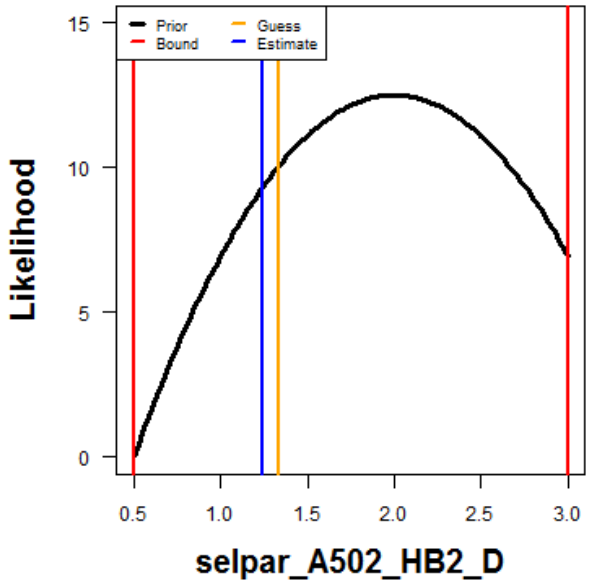

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_HB3_D

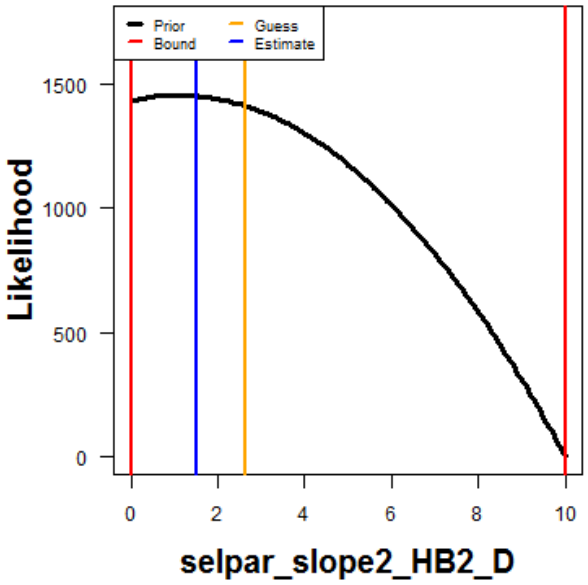

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_HB3_D

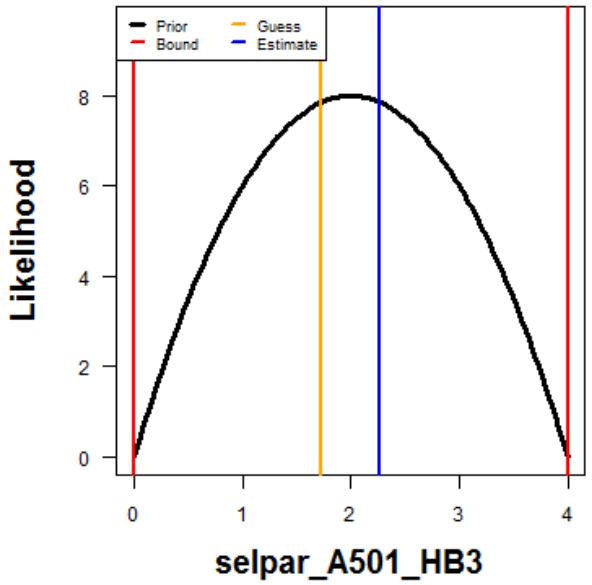

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB3_D

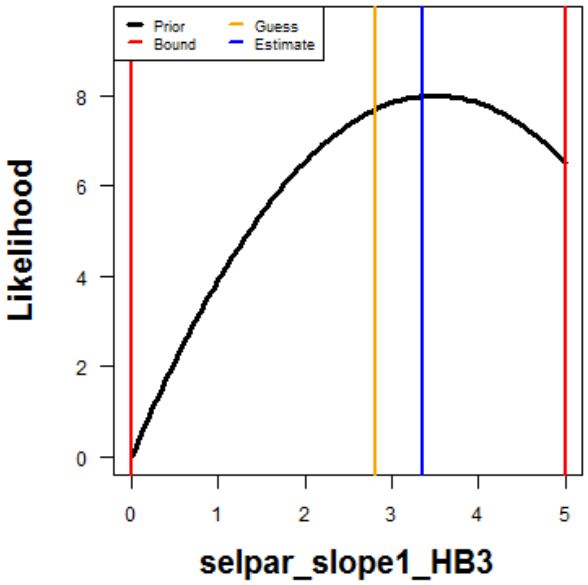

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_HB2_D

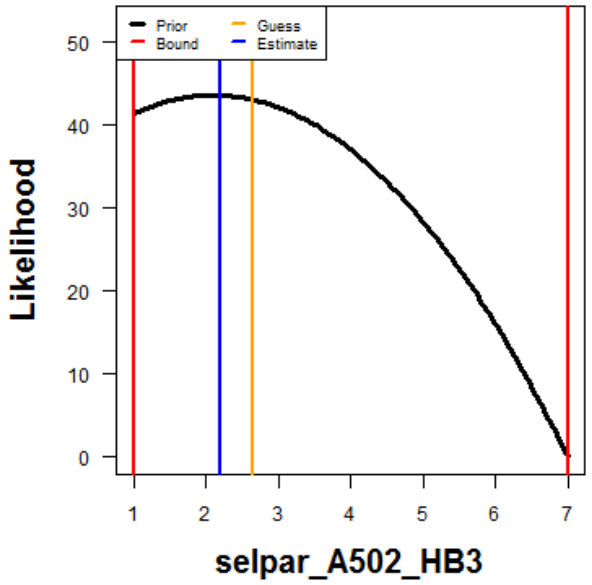

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_HB2_D

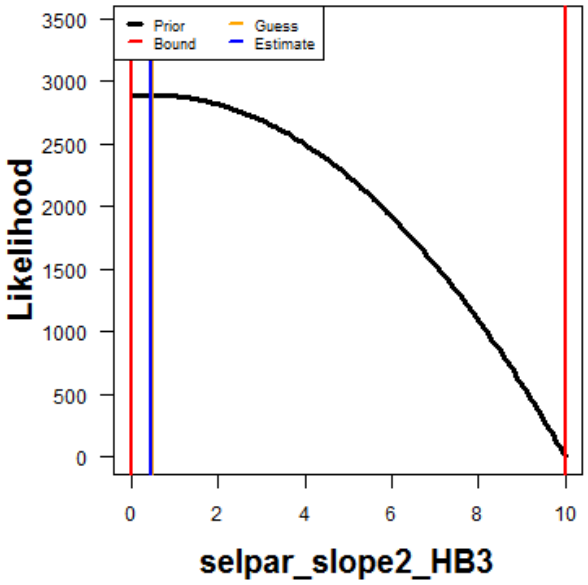

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_HB2_D

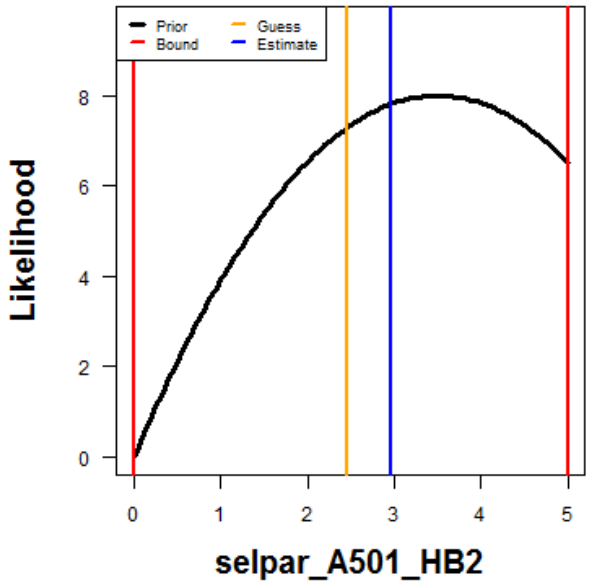

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB2_D

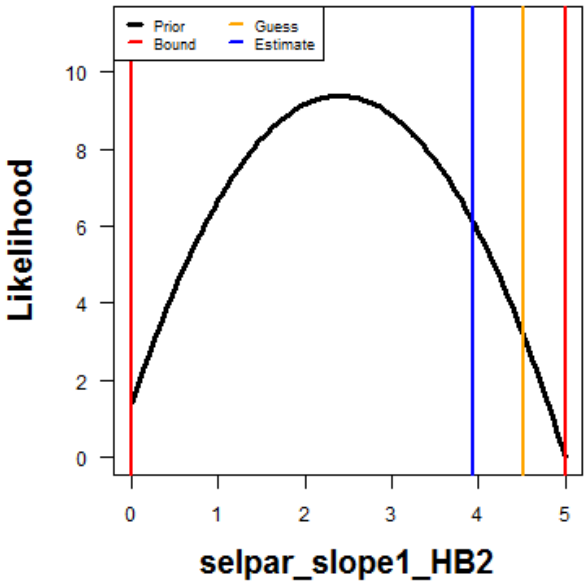

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_HB3

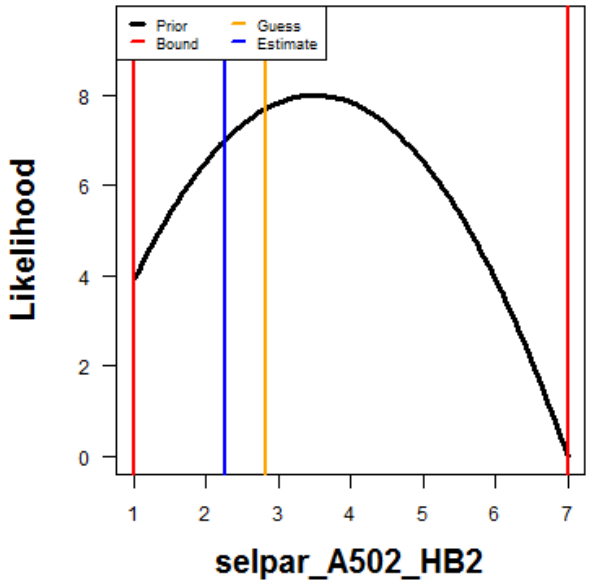

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_HB3

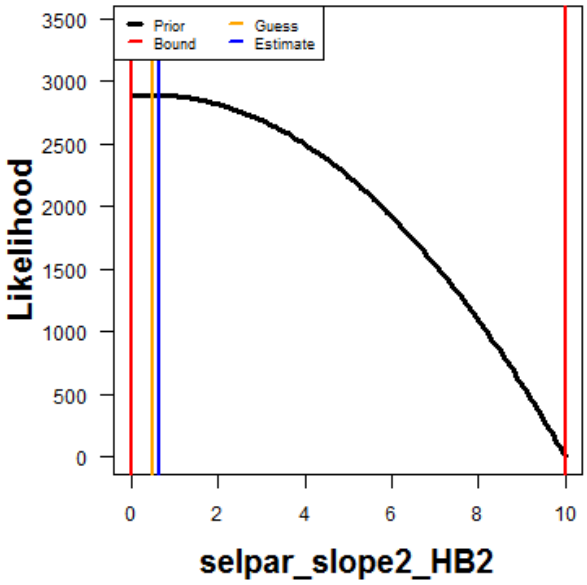

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_HB3

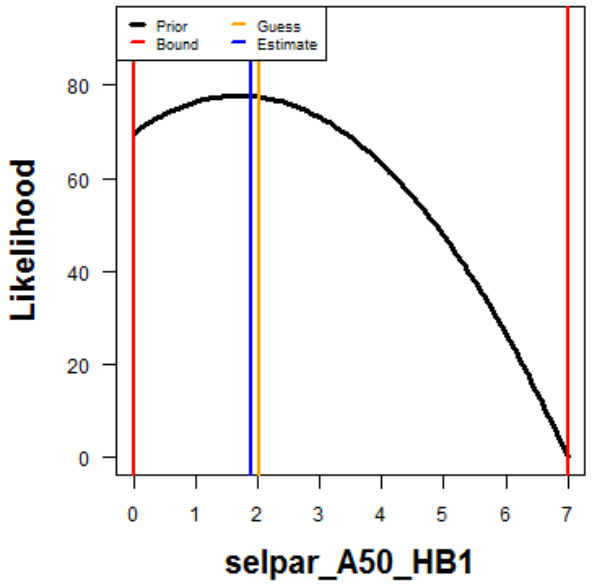

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB3

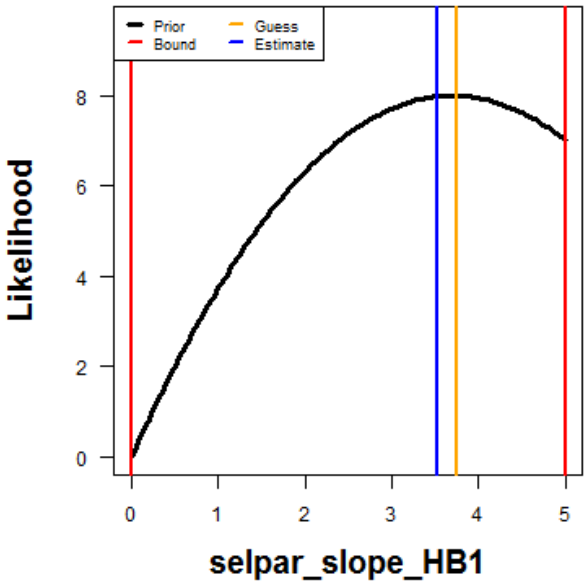

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_HB2

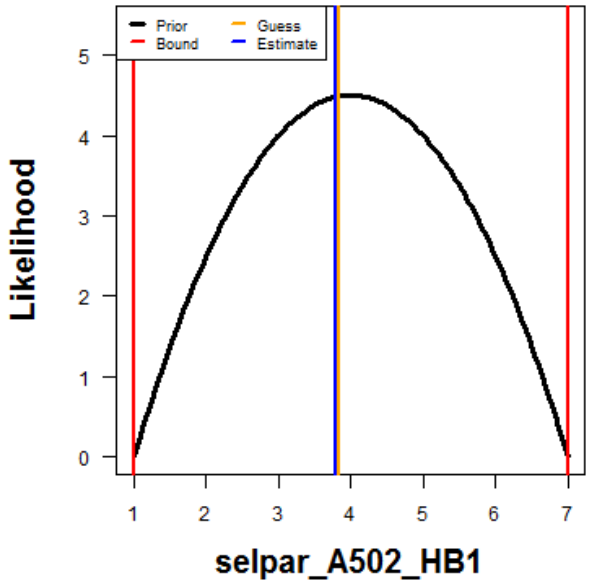

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_HB2

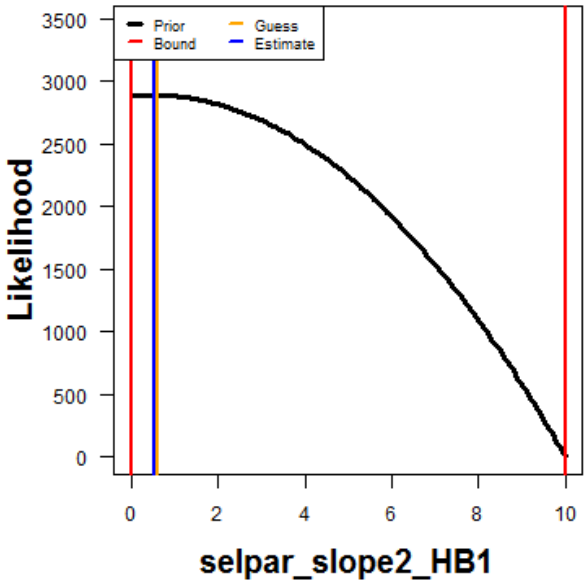

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_HB2

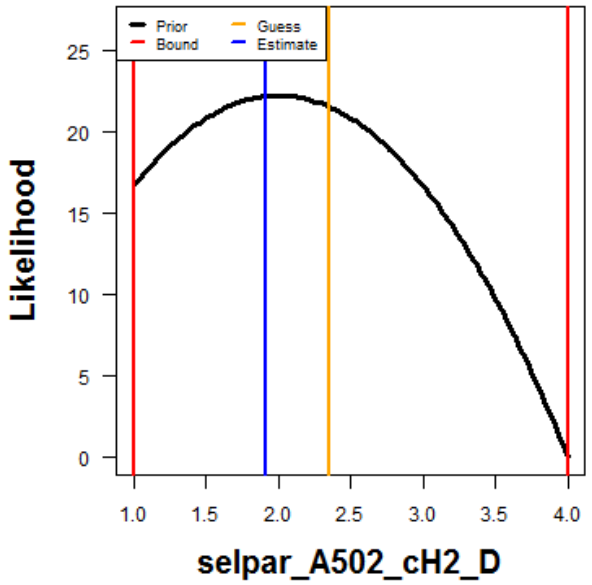

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB2

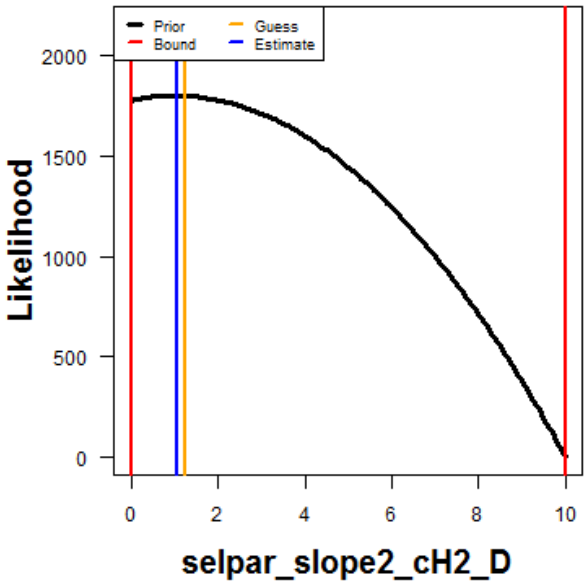

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_HB1

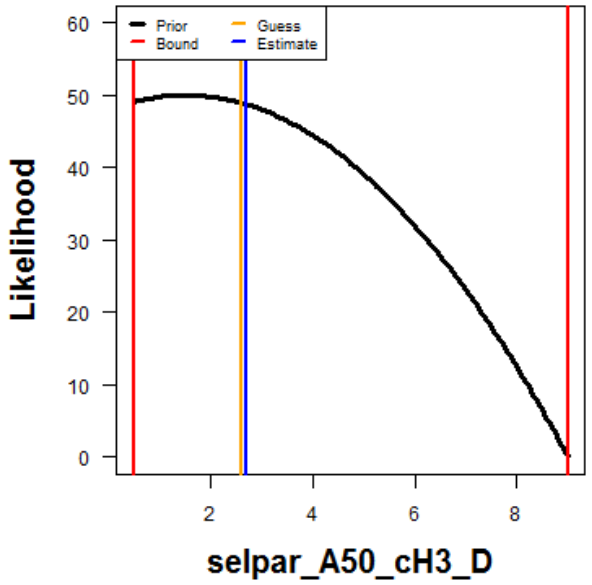

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_HB1

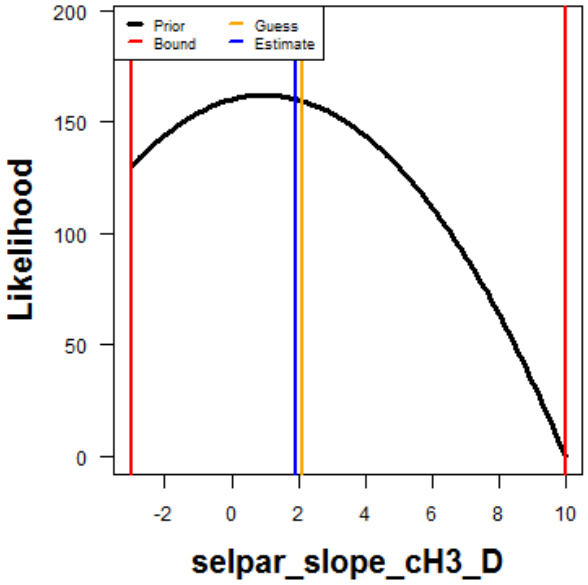

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_HB1

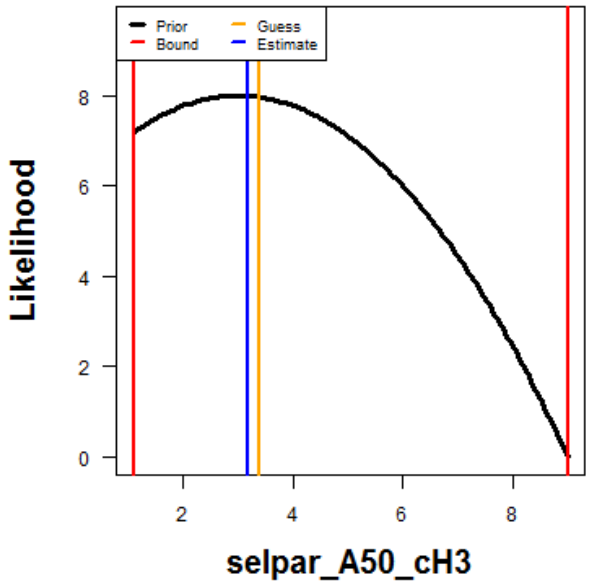

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_HB1

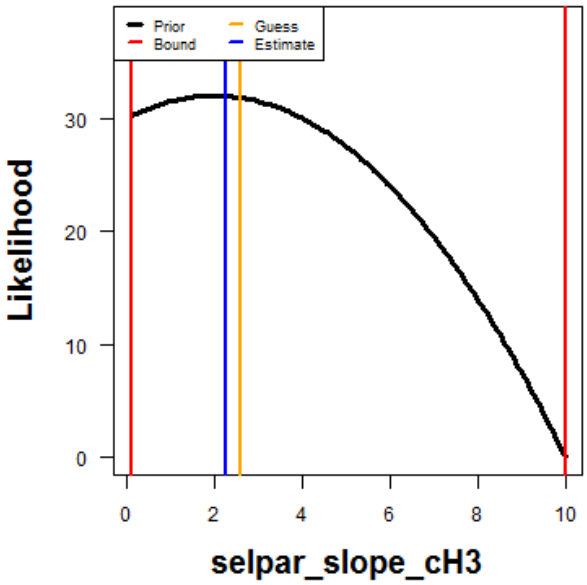

Prior
Bound
Guess
Estimate
Likelihood
selpar_A502_cH2_D

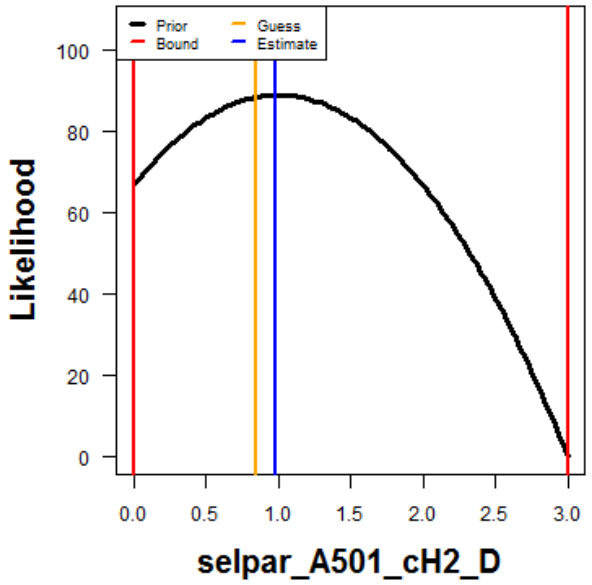

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope2_cH2_D

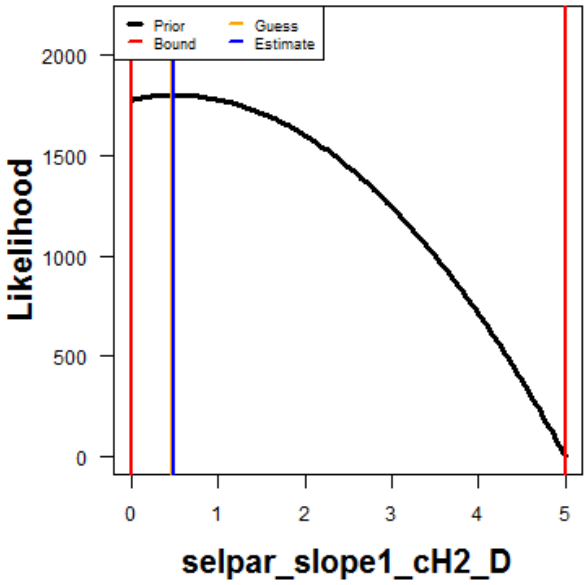

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_cH3_D

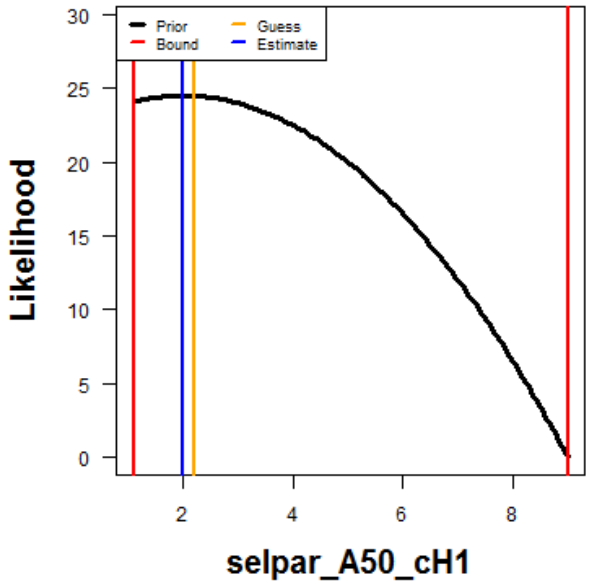

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_cH3_D

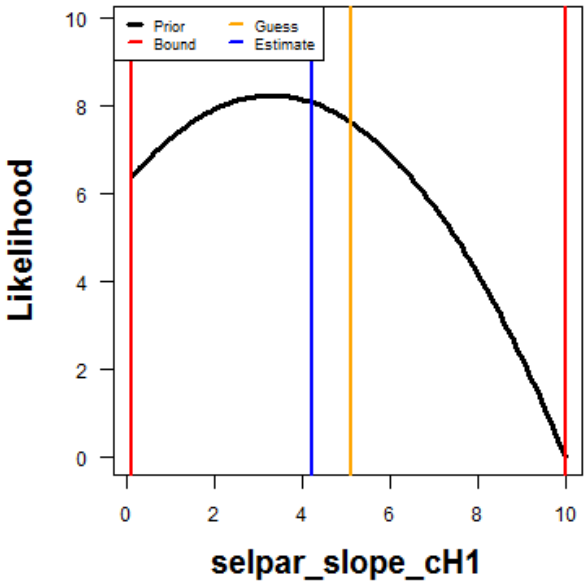

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_cH3

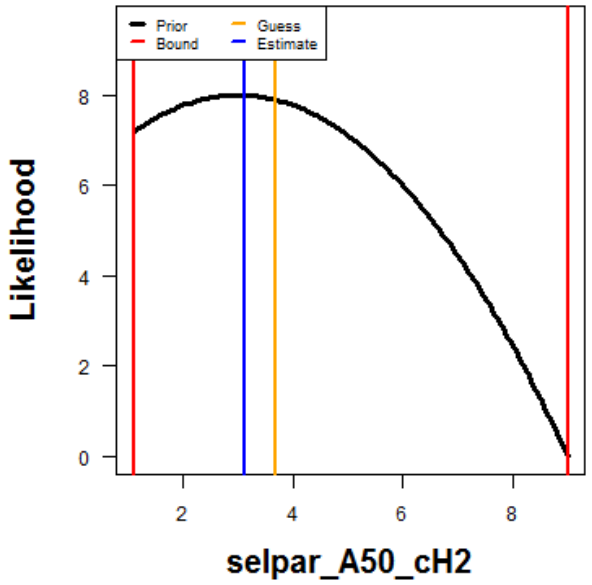

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_cH3

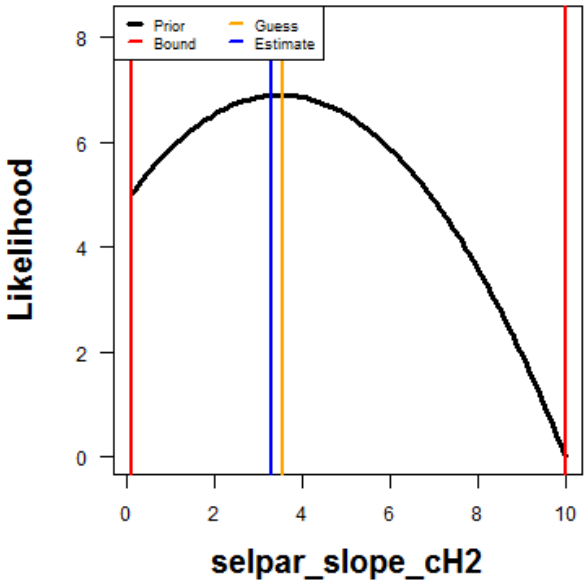

Prior
Bound
Guess
Estimate
Likelihood
selpar_A501_cH2_D

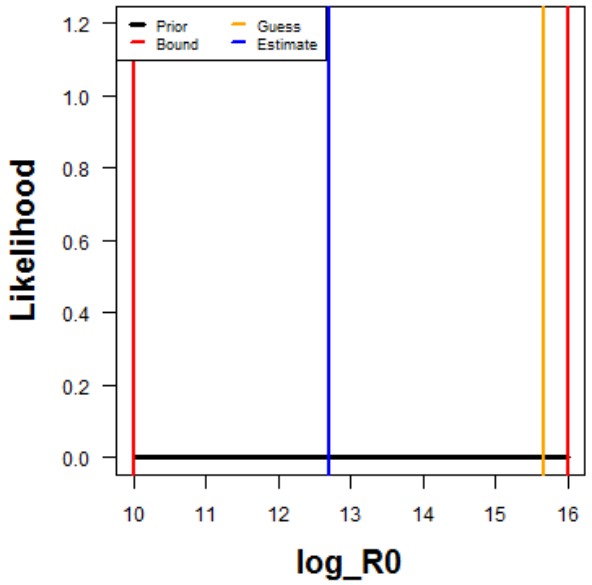

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope1_cH2_D

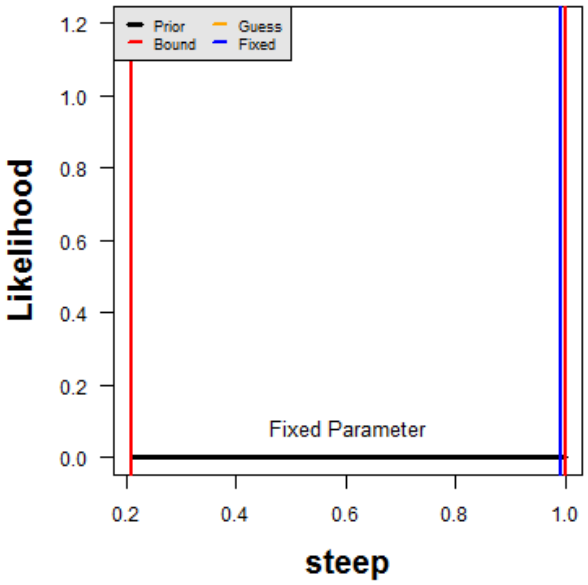

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_cH1

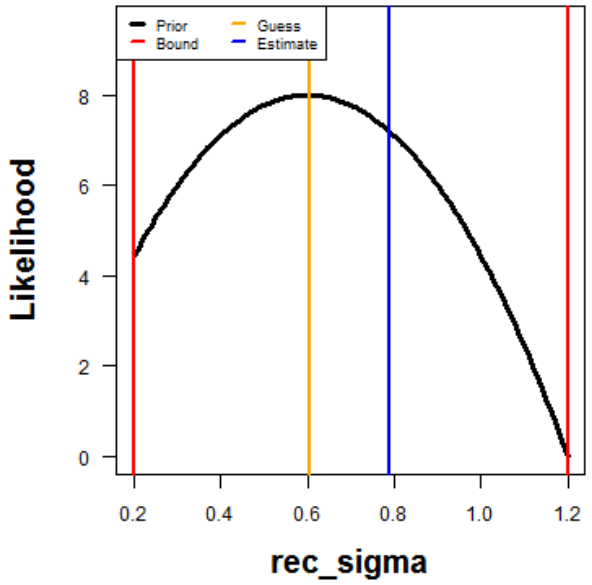

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_cH1

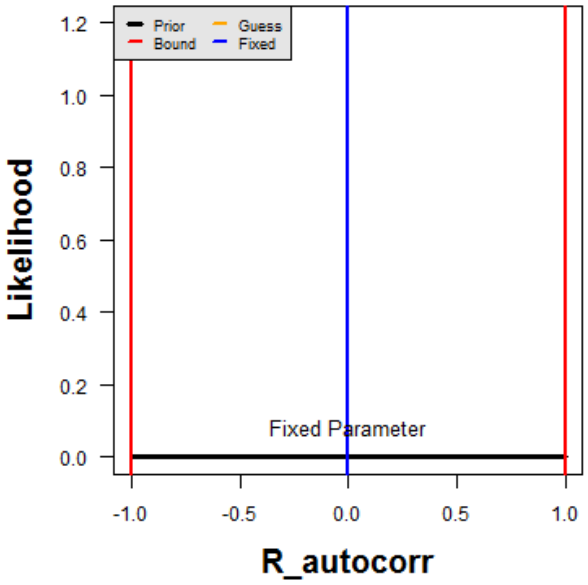

Prior
Bound
Guess
Estimate
Likelihood
selpar_A50_cH2

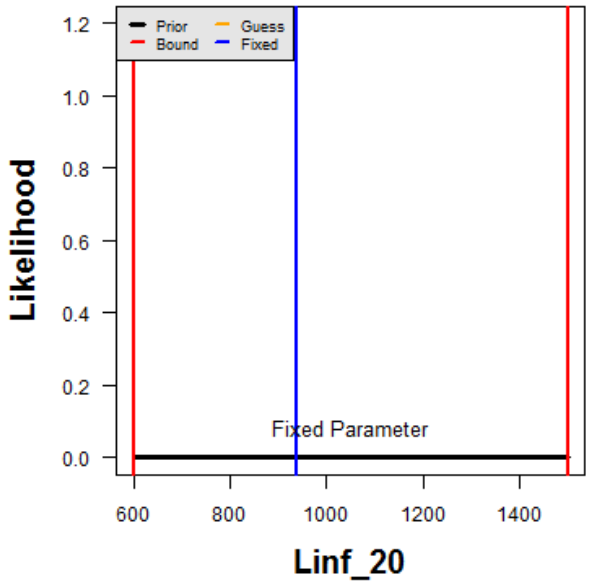

Prior
Bound
Guess
Estimate
Likelihood
selpar_slope_cH2

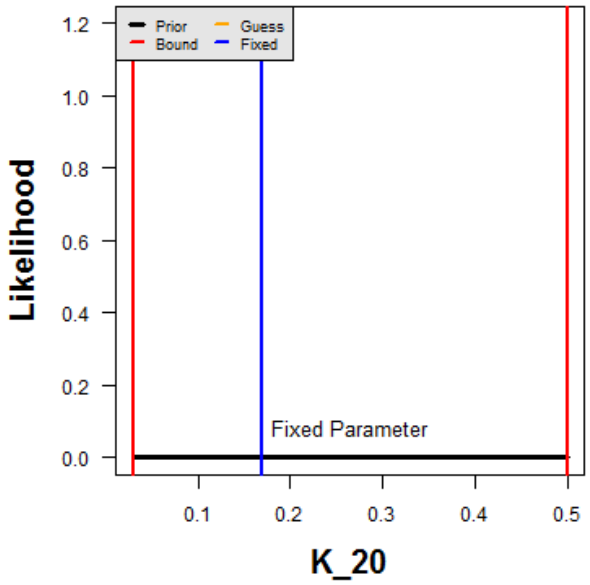

Prior
Bound
Guess
Estimate
Likelihood
log_R0

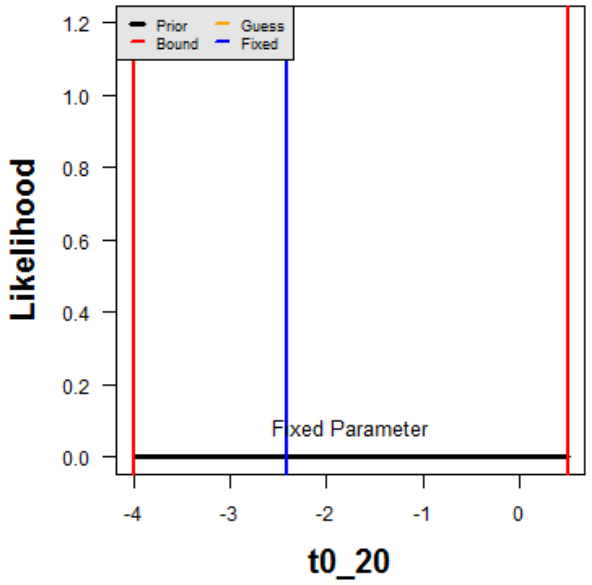

Prior
Bound
Guess
Fixed
Likelihood
Fixed Parameter
steep

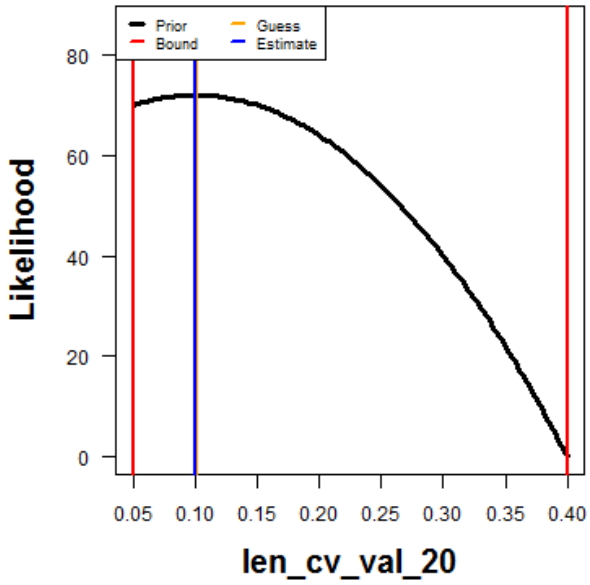

Prior
Bound
Guess
Estimate
Likelihood
rec_sigma

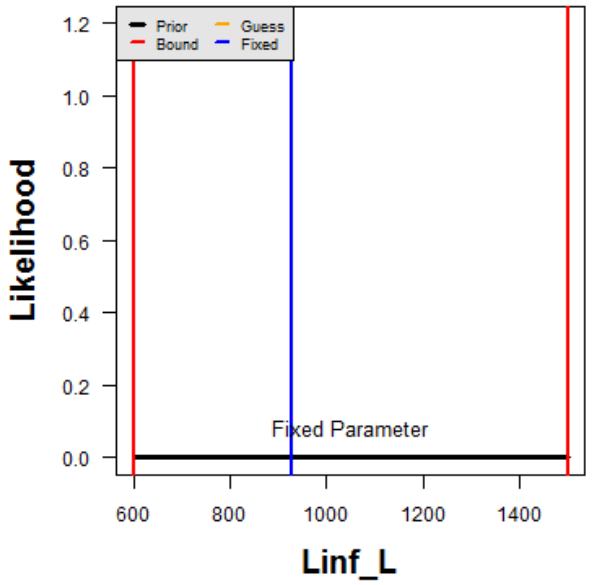

Prior
Bound
Guess
Fixed
Likelihood
Fixed Parameter
R_autocorr

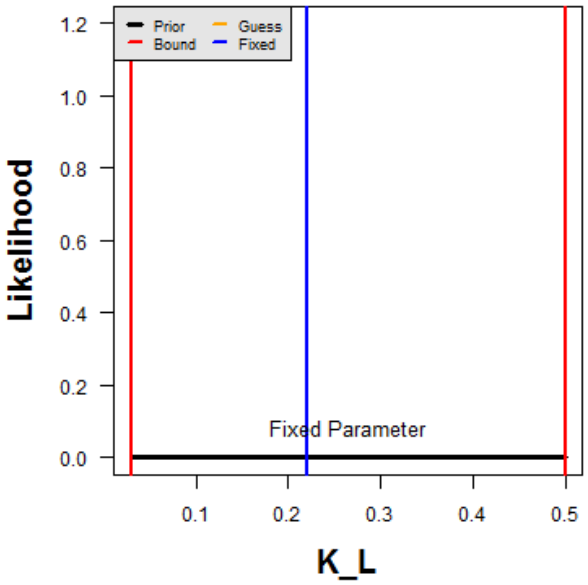

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
Linf_20

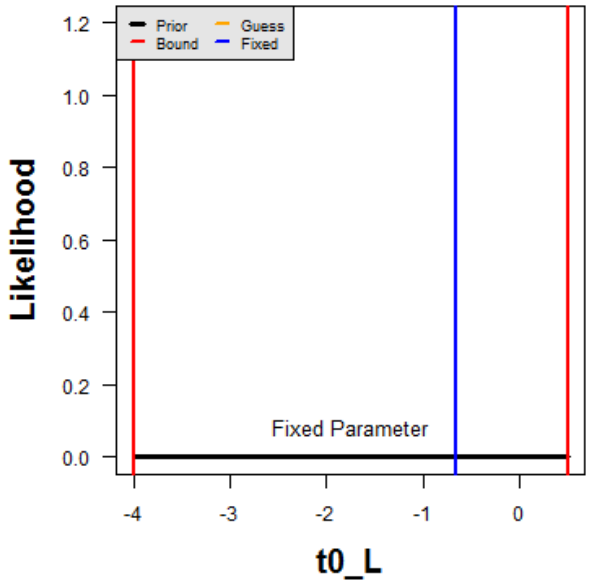

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
K_20

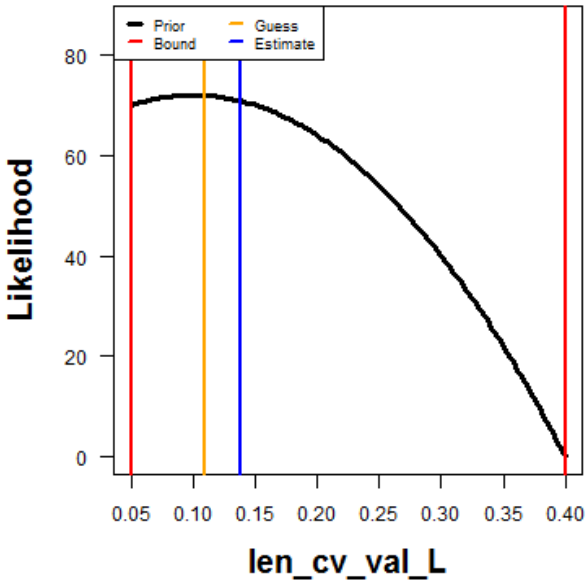

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
t0_20

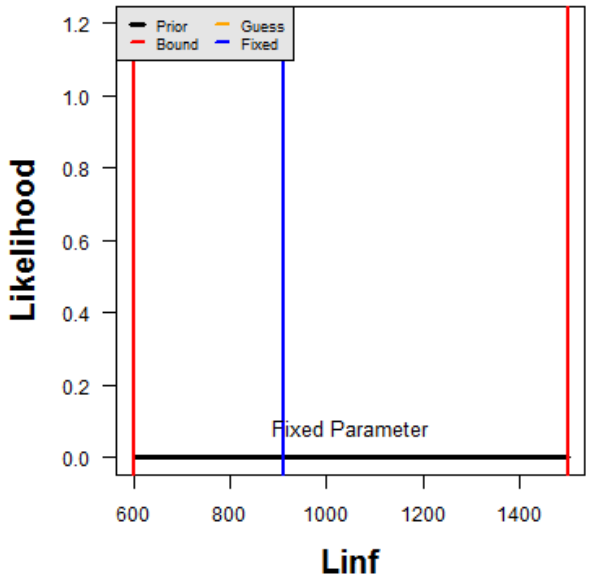

Prior
Guess
Bound
Estimate
Likelihood
len_cv_val_20

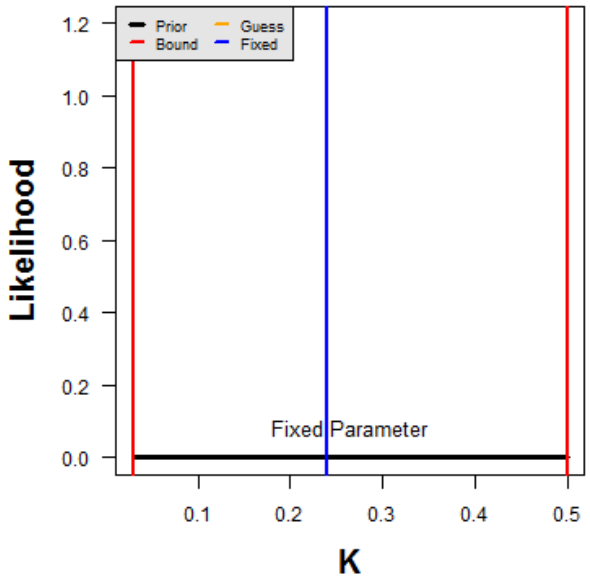

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
Linf_L

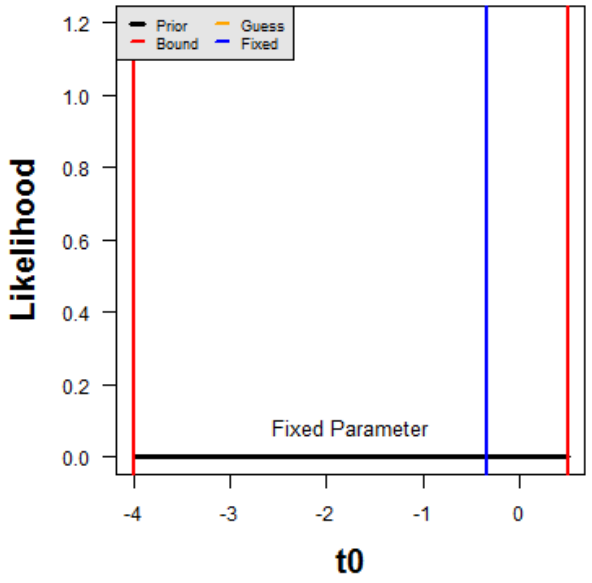

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
K_L

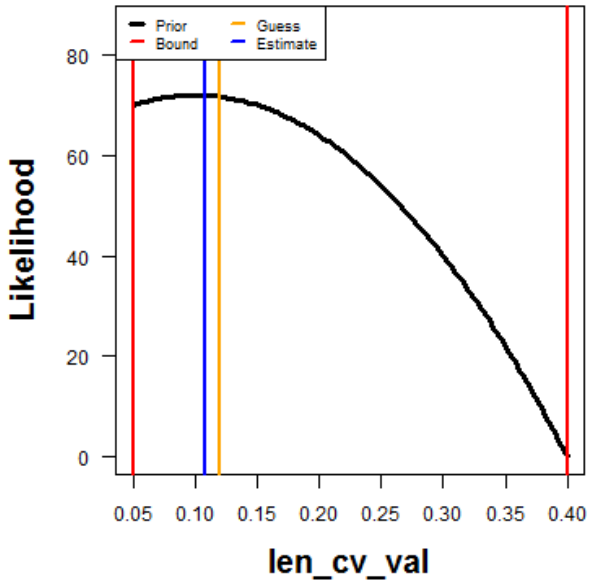

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
t0_L

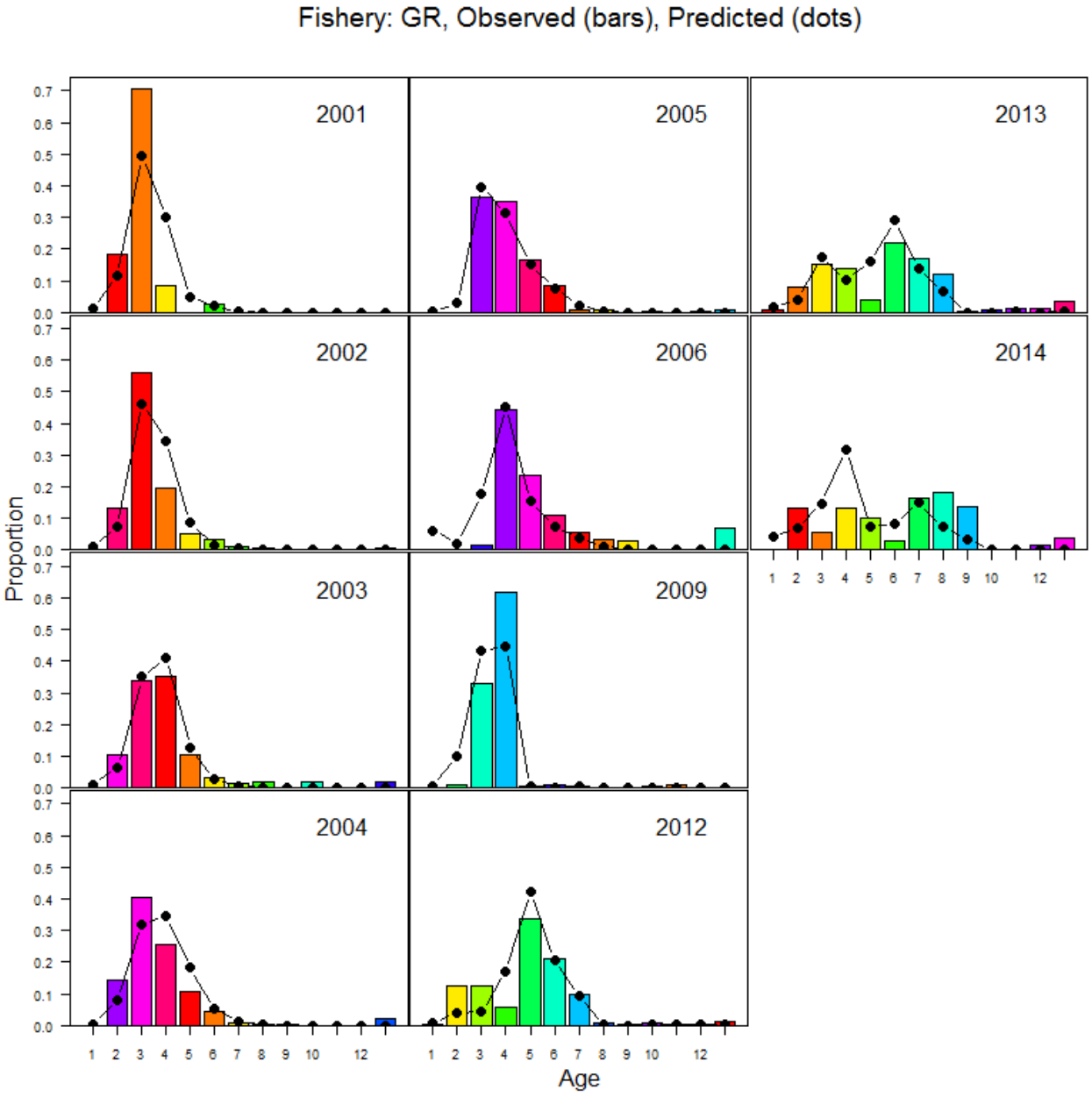

Prior
Guess
Bound
Estimate
Likelihood
len_cv_val_L

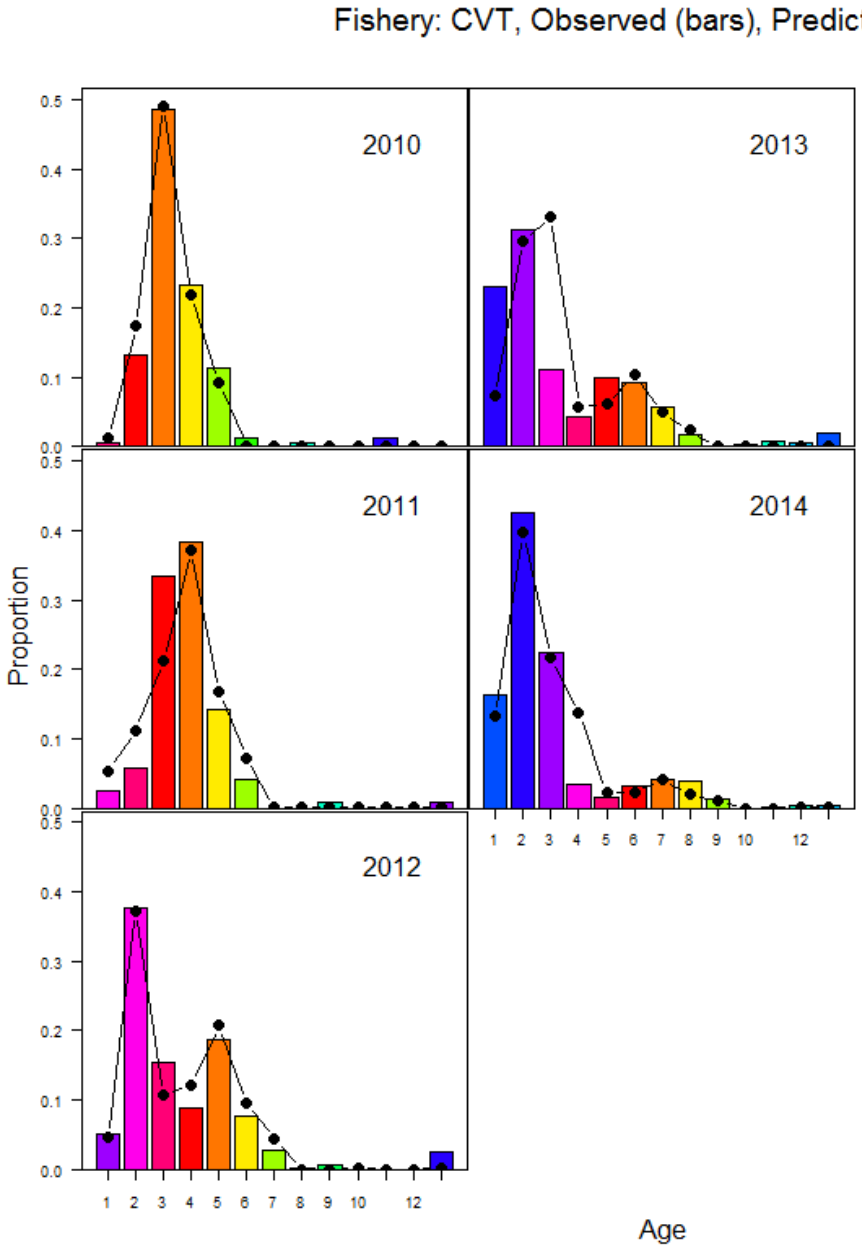

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
Linf

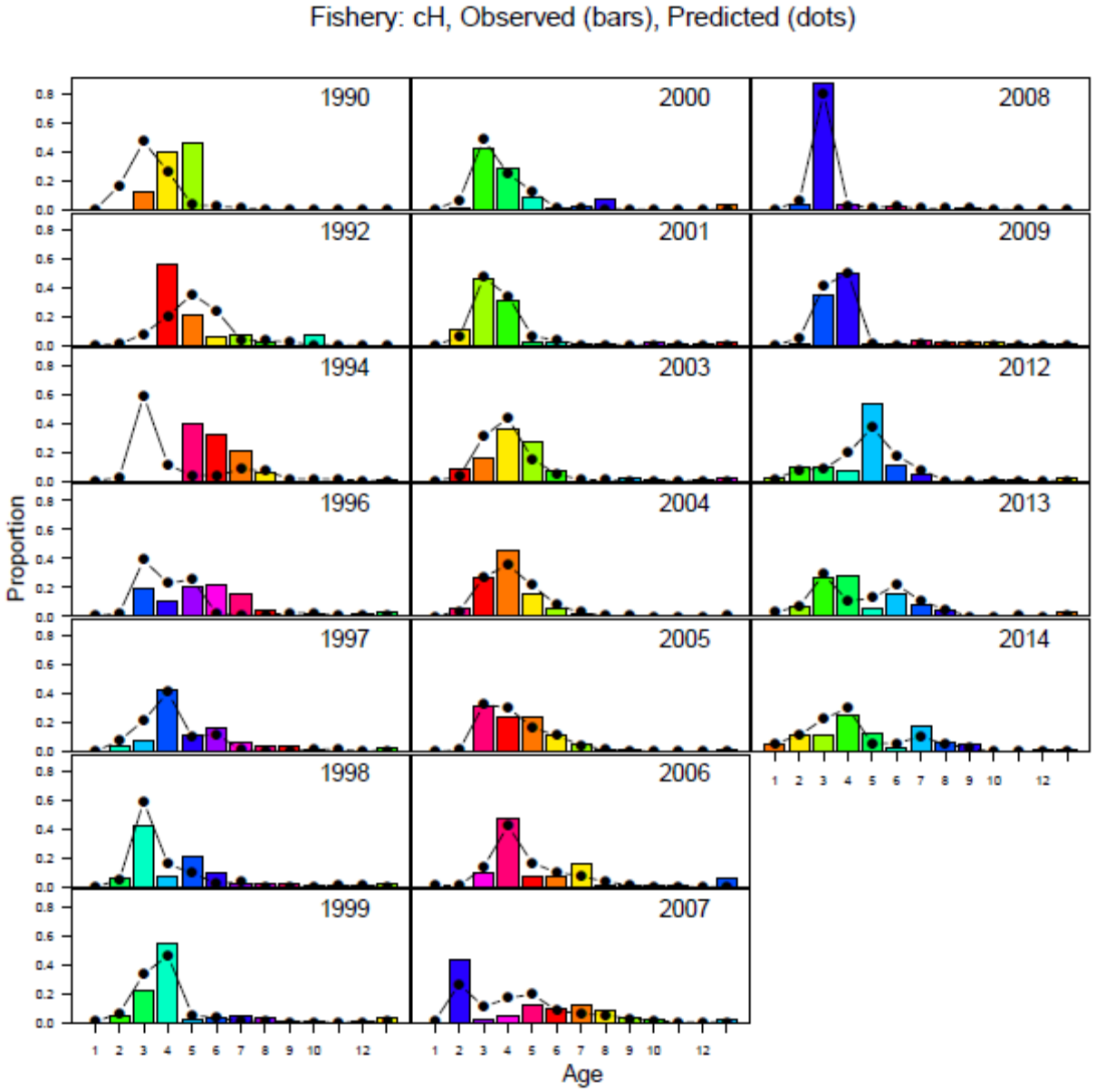

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
K

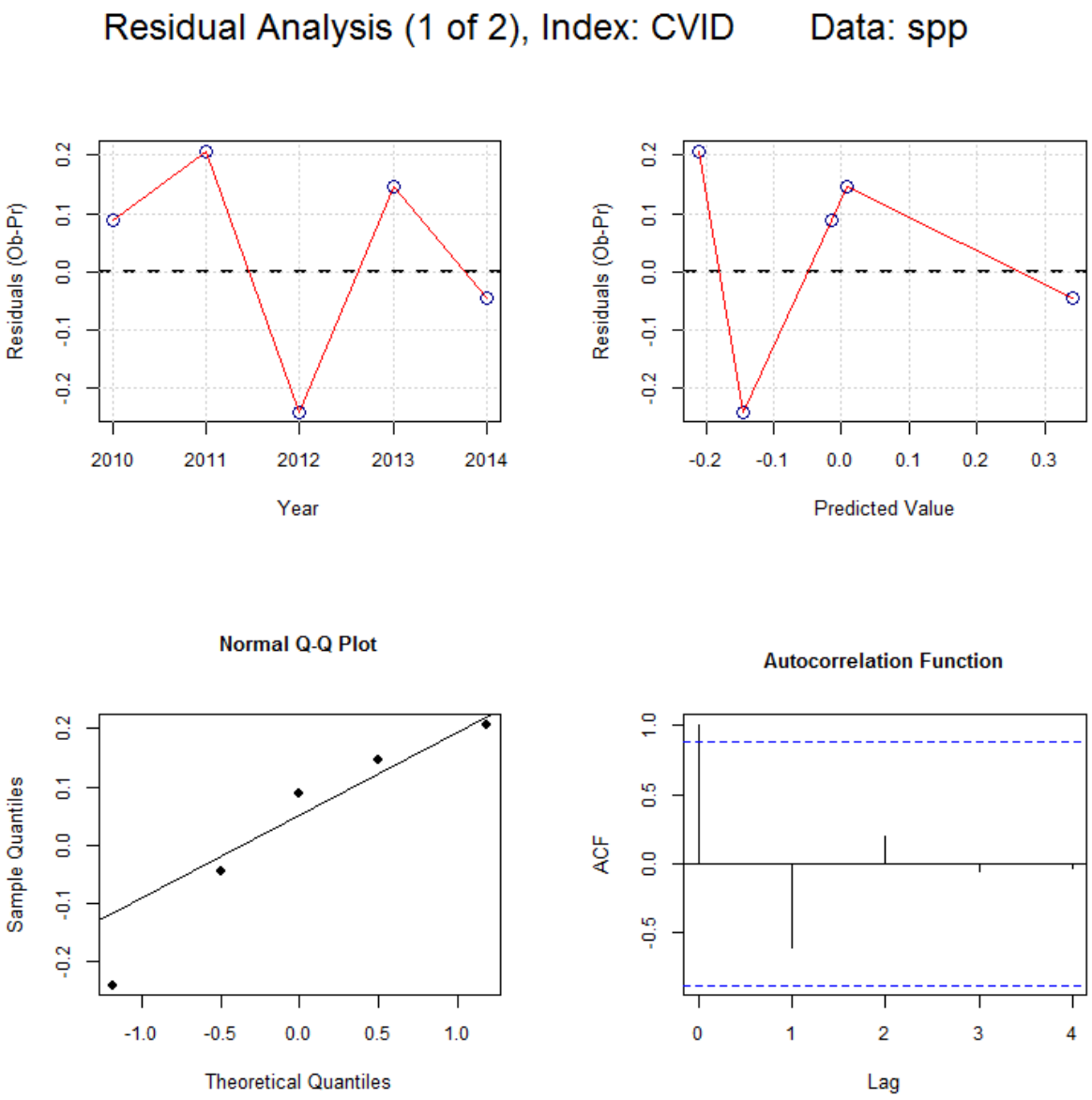

Prior
Guess
Bound
Fixed
Likelihood
Fixed Parameter
t0

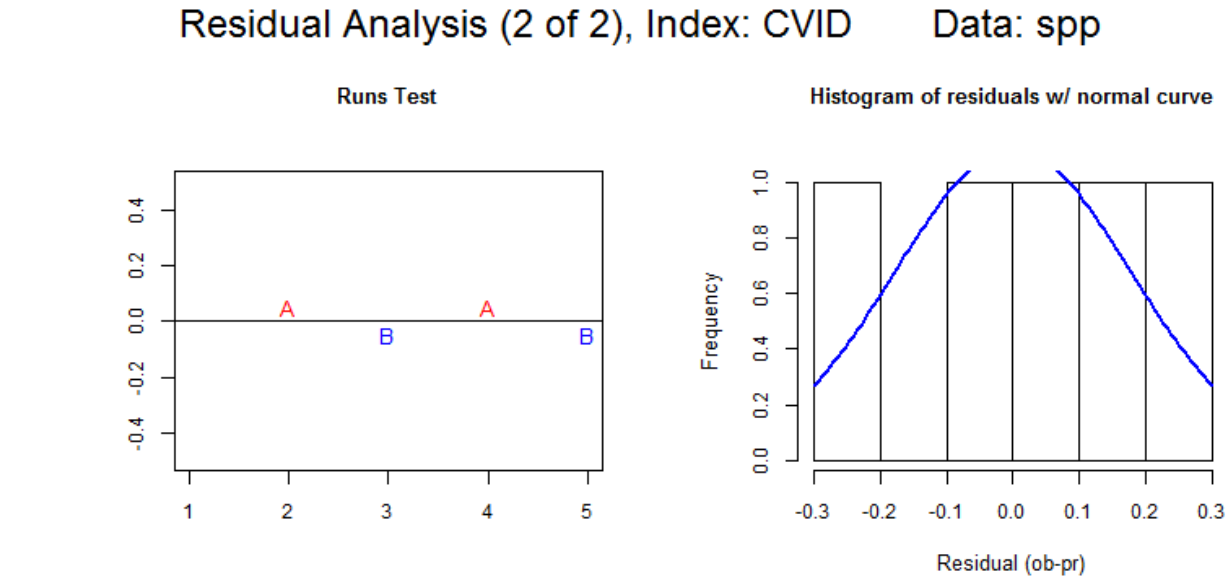

Prior
Guess
Bound
Estimate
Likelihood
len_cv_val


Fishery: GR, Observed (bars), Predicted (dots)
2001
2005
2013
2002
2006
2014
2003
2009
2004
2012
Proportion
Age

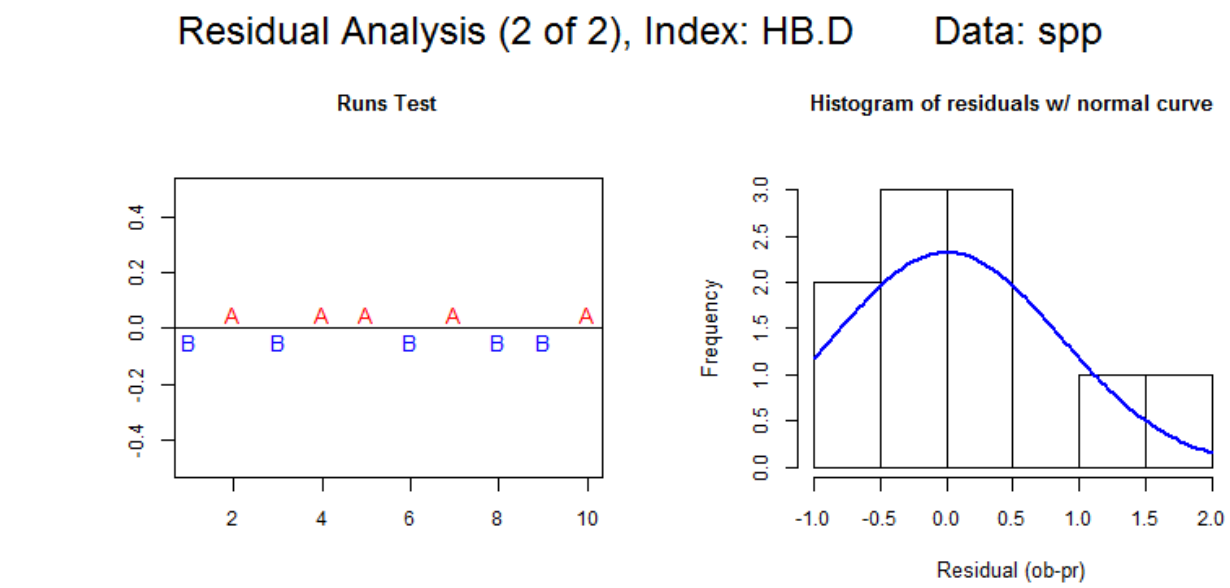

Fishery: CVT, Observed (bars), Predicted (dots)
2010
2013
2011
2014
2012
Proportion
Age


Fishery: cH, Observed (bars), Predicted (dots)
Proportion
Age
1990
2000
2008
1992
2001
2009
1994
2003
2012
1996
2004
2013
1997
2005
2014
1998
2006
1999
2007

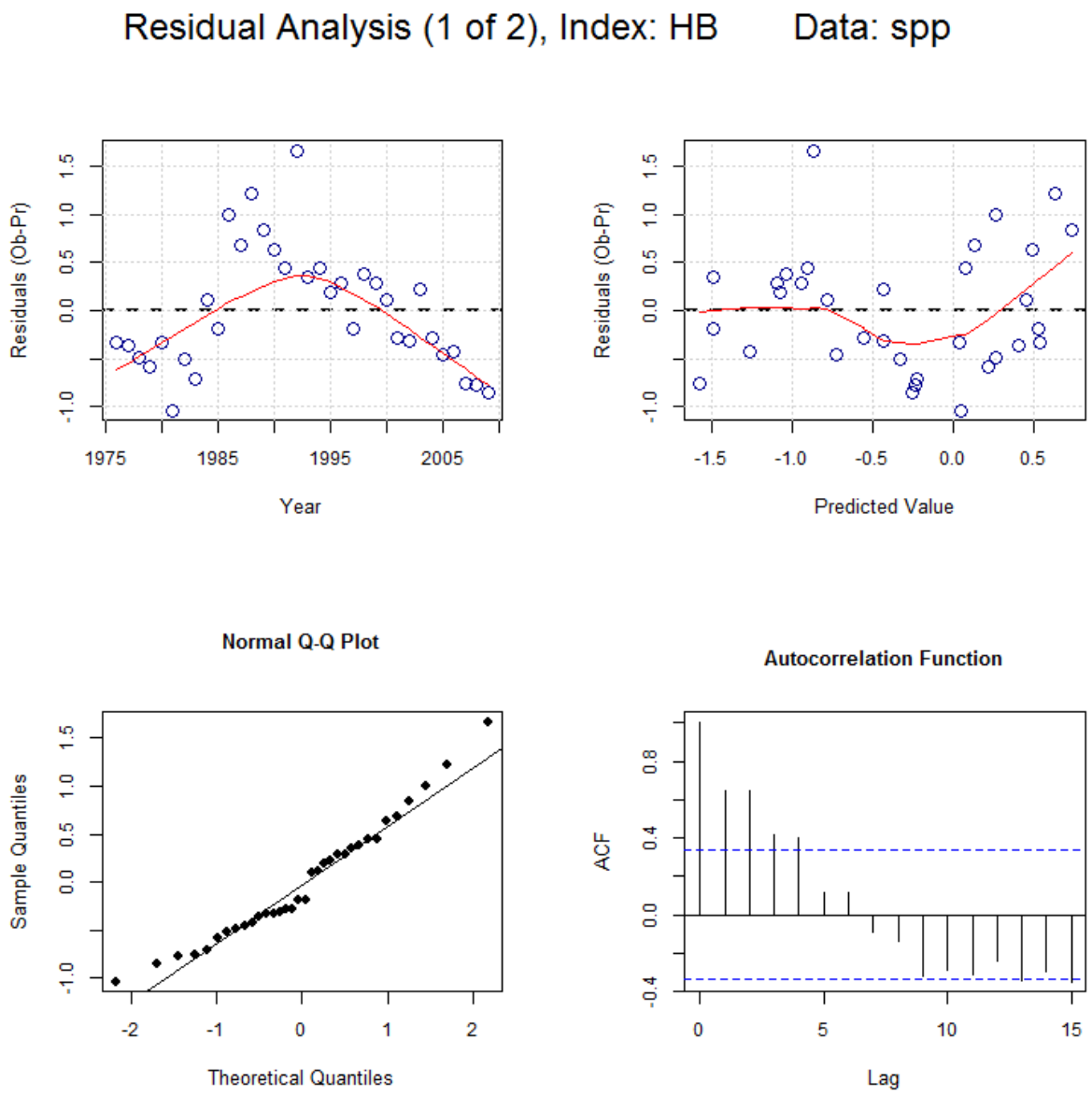

Residual Analysis (1 of 2), Index: CVID Data: spp
Residuals (Ob-Pr)
Year
Predicted Value
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Autocorrelation Function
ACF
Lag

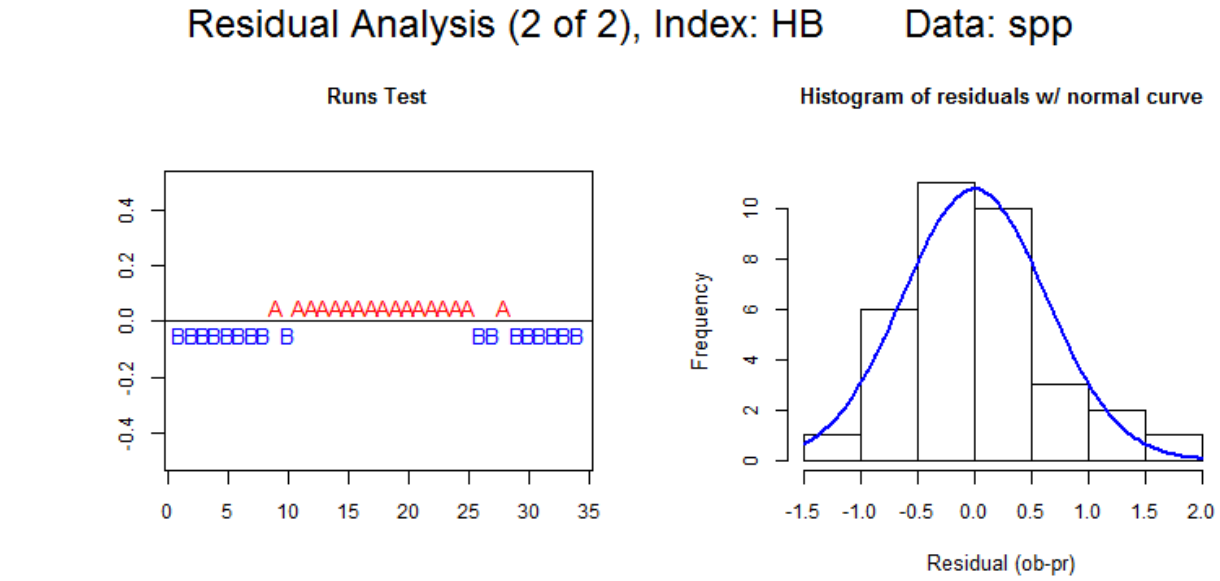


	P-values
Breusch-Pagan test for heteroskedasticity:	0.8917
Harrison-McCabe test for heteroskedasticity:	0.555
Breusch-Godfrey test for higher-order serial correlation:	0.3117
Durbin-Watson test for autocorrelation of disturbances:	0.0195
Lilliefors (Kolmogorov-Smirnov) test for normality:	0.5293
Anderson-Darling test for normality:	NA
Pearson chi-square test for normality:	0.4386
Shapiro-Wilk test for normality:	0.5742
Phillips-Perron test for null hypothesis x has a unit root:	0.713
Runs test:	0.0143

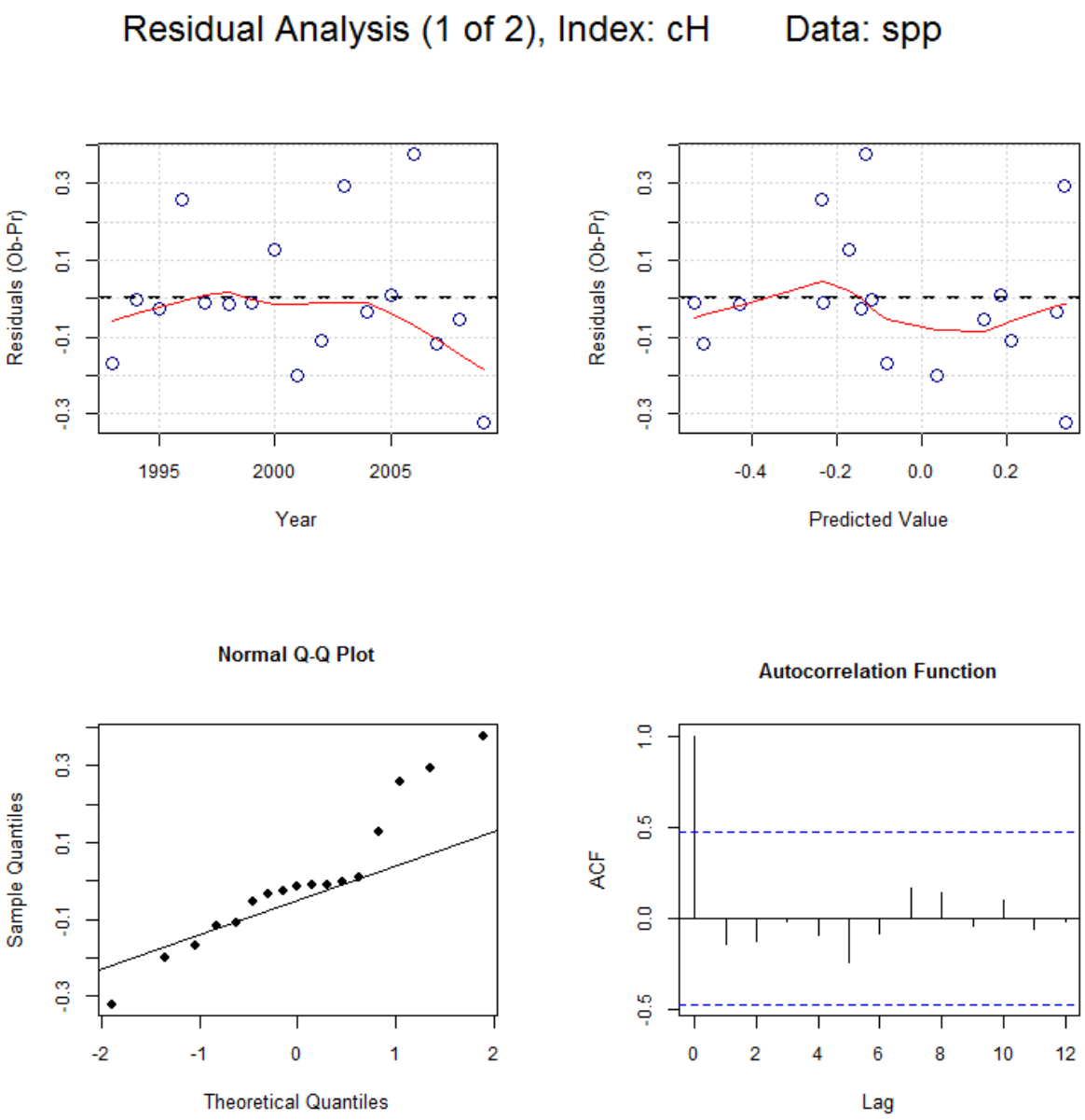


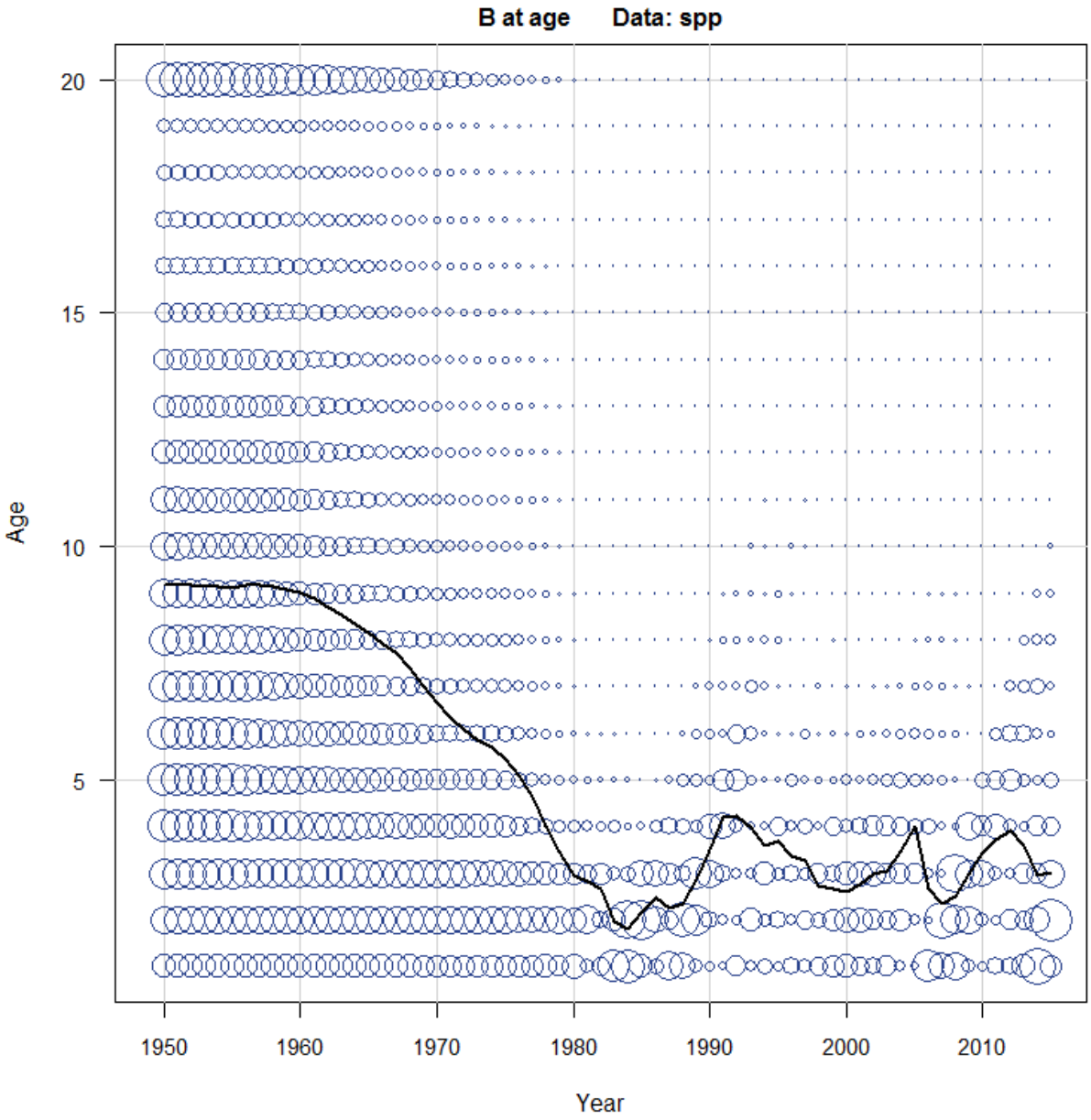
	P-values
Breusch-Pagan test for heteroskedasticity:	0.5039
Harrison-McCabe test for heteroskedasticity:	0.524
Breusch-Godfrey test for higher-order serial correlation:	0.0077
Durbin-Watson test for autocorrelation of disturbances:	0
Lilliefors (Kolmogorov-Smirnov) test for normality:	0.7478
Anderson-Darling test for normality:	0.472
Pearson chi-square test for normality:	0.3618
Shapiro-Wilk test for normality:	0.3786
Phillips-Perron test for null hypothesis x has a unit root:	0.2968
Runs test:	0.1797

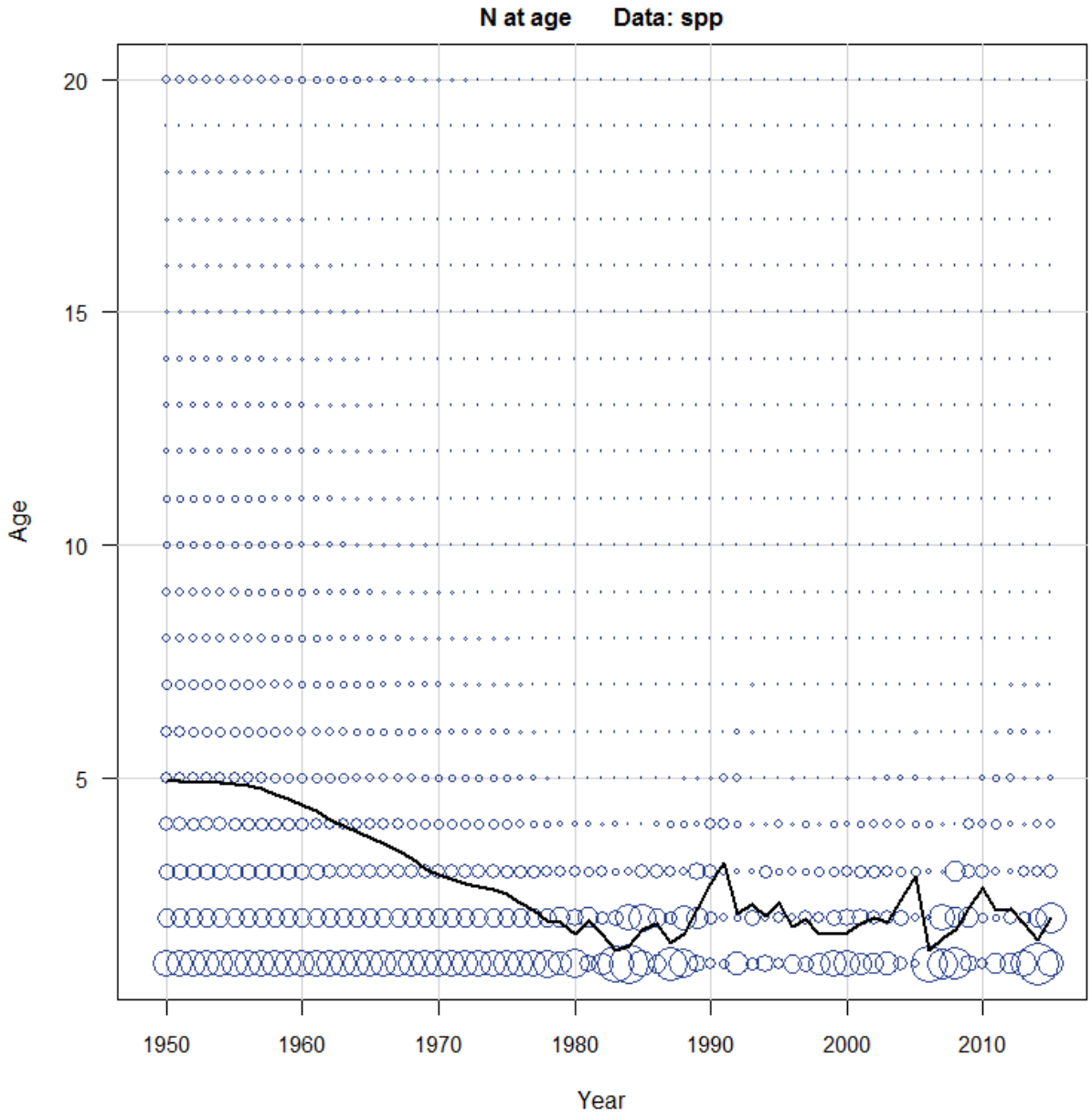

Residual Analysis (1 of 2), Index: HB Data: spp
Residuals (Ob-Pr)
Year
Predicted Value
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Autocorrelation Function
ACF
Lag

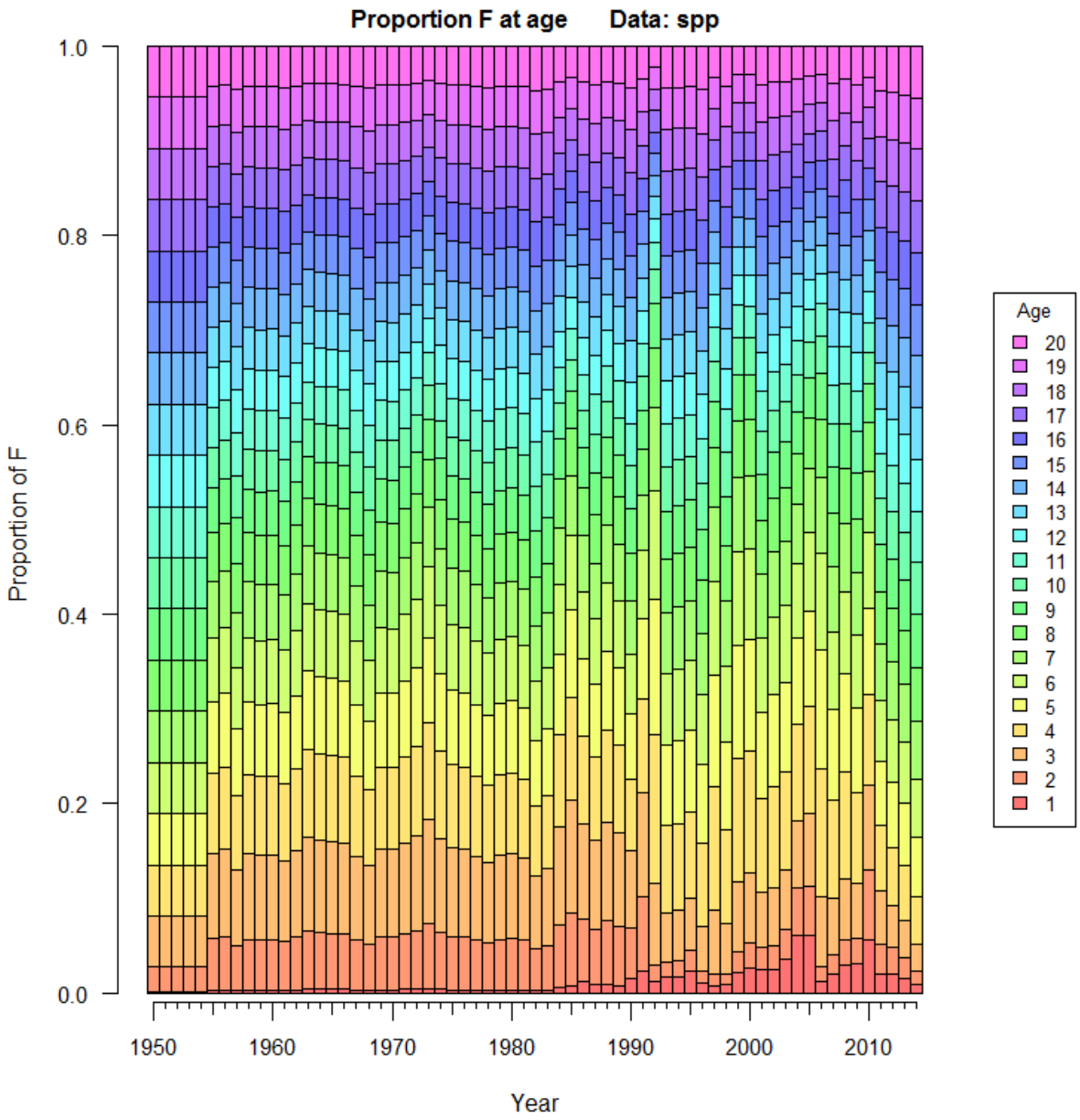


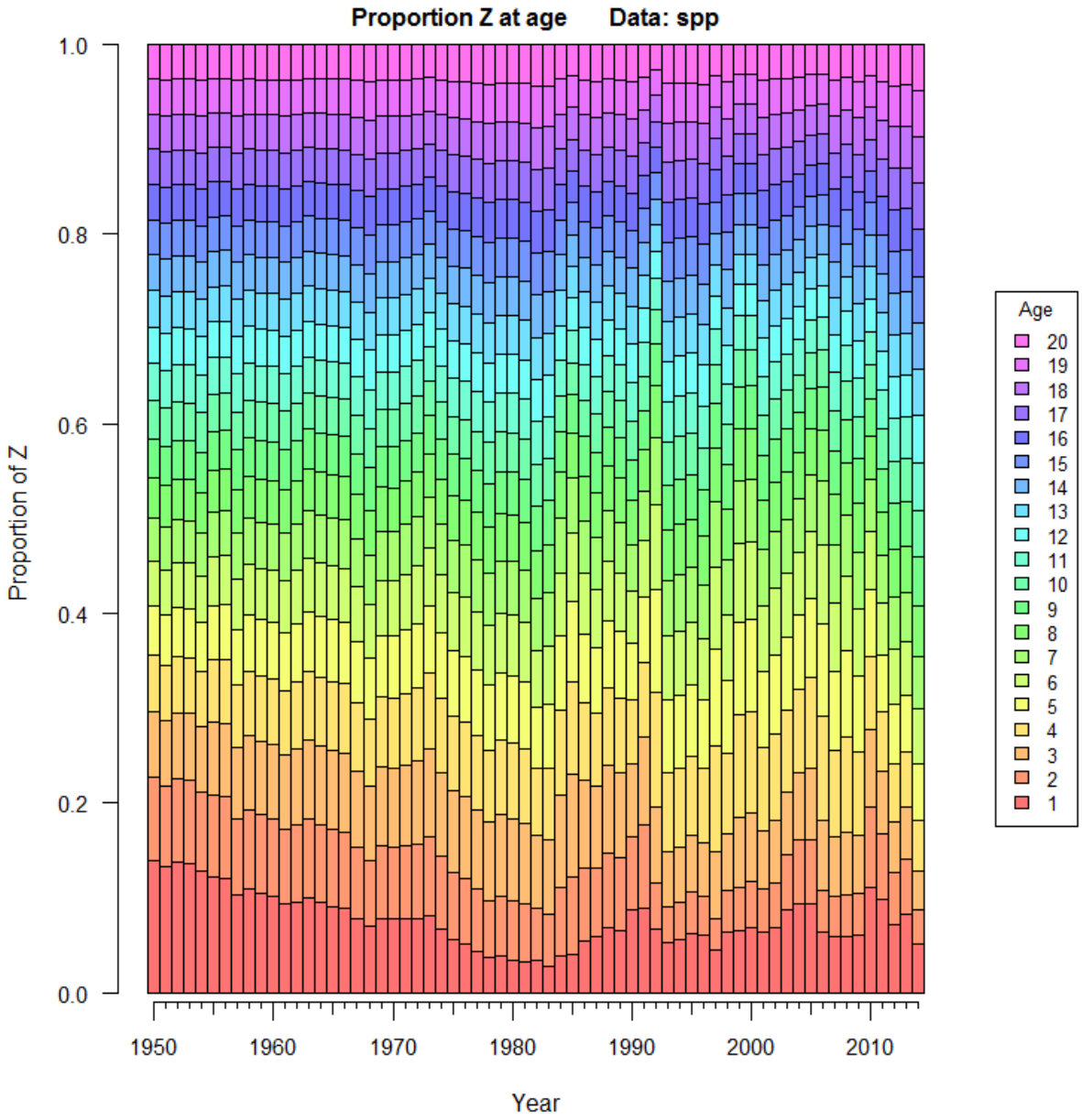
	P-values
Breusch-Pagan test for heteroskedasticity:	0
Harrison-McCabe test for heteroskedasticity:	0.612
Breusch-Godfrey test for higher-order serial correlation:	0
Durbin-Watson test for autocorrelation of disturbances:	0
Lilliefors (Kolmogorov-Smirnov) test for normality:	0.0703
Anderson-Darling test for normality:	0.2309
Pearson chi-square test for normality:	0.0607
Shapiro-Wilk test for normality:	0.2479
Phillips-Perron test for null hypothesis x has a unit root:	0.5759
Runs test:	1e-04

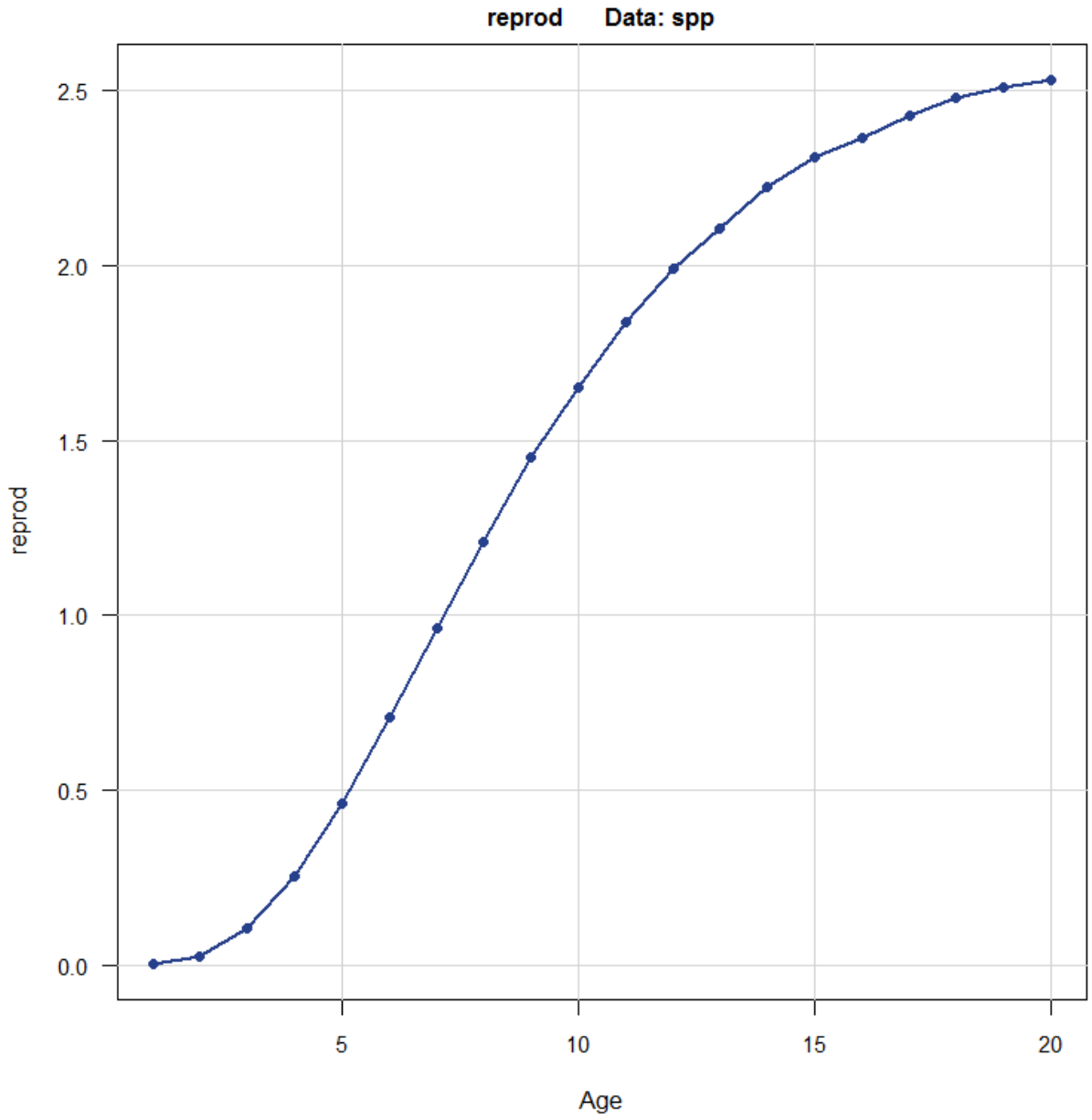

Residual Analysis (1 of 2), Index: cH Data: spp
Residuals (Ob-Pr)
Year
Predicted Value
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Autocorrelation Function
ACF
Lag

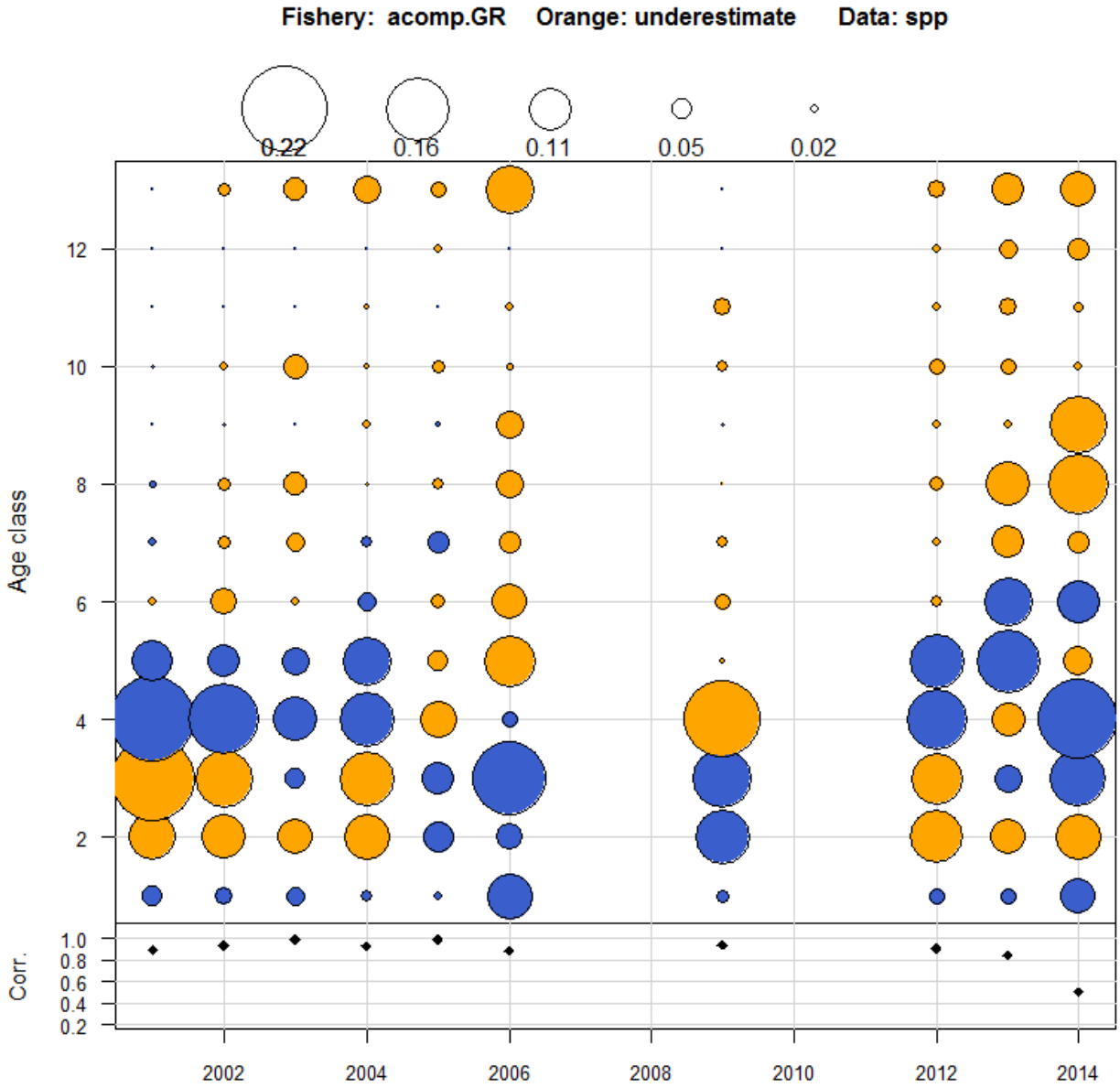

B at age Data: spp
Age
20
15
10
5
1950
1960
1970
1980
1990
2000
2010
Year

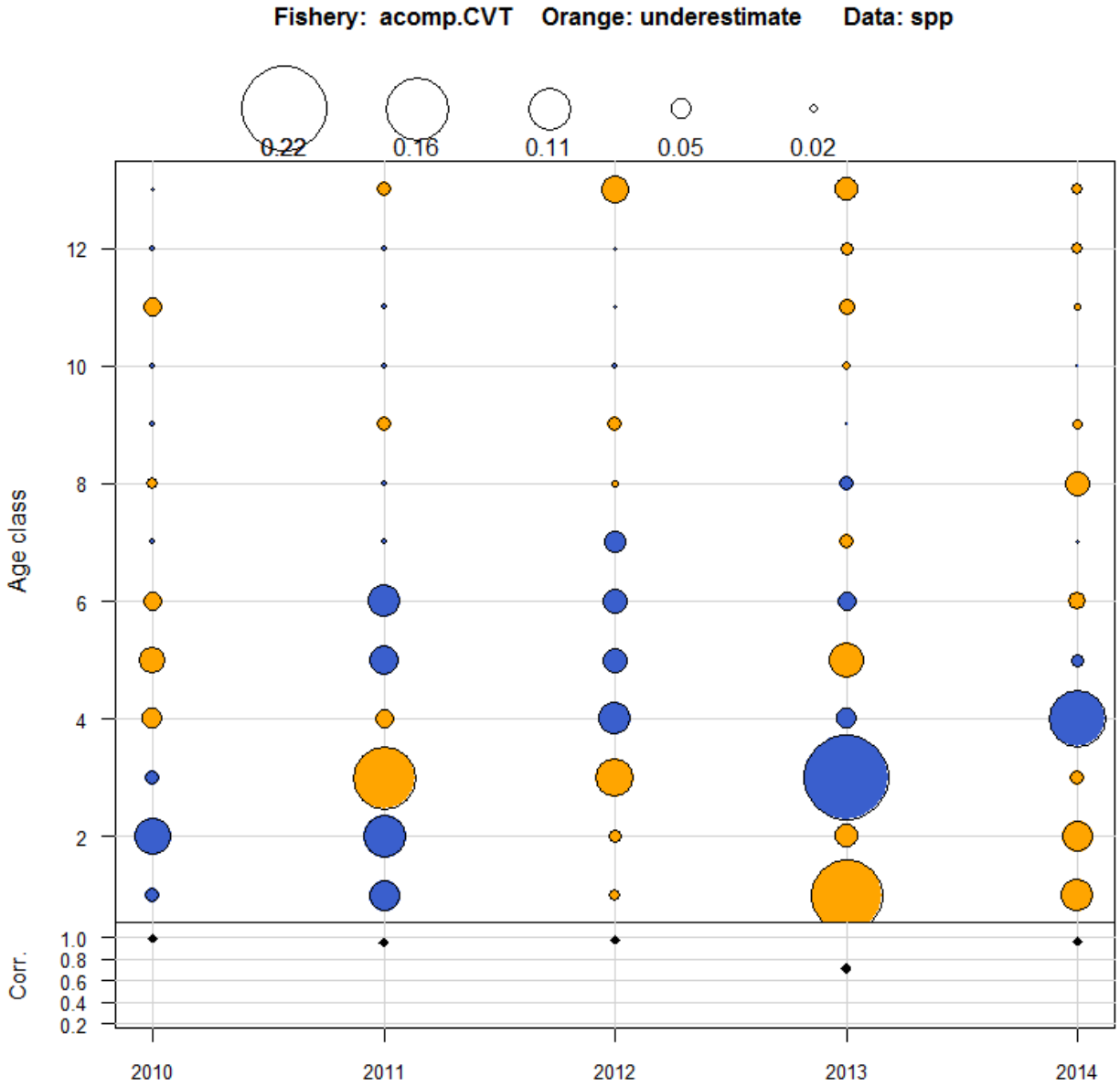

N at age Data: spp
Age
20
15
10
5
1950
1960
1970
1980
1990
2000
2010
Year

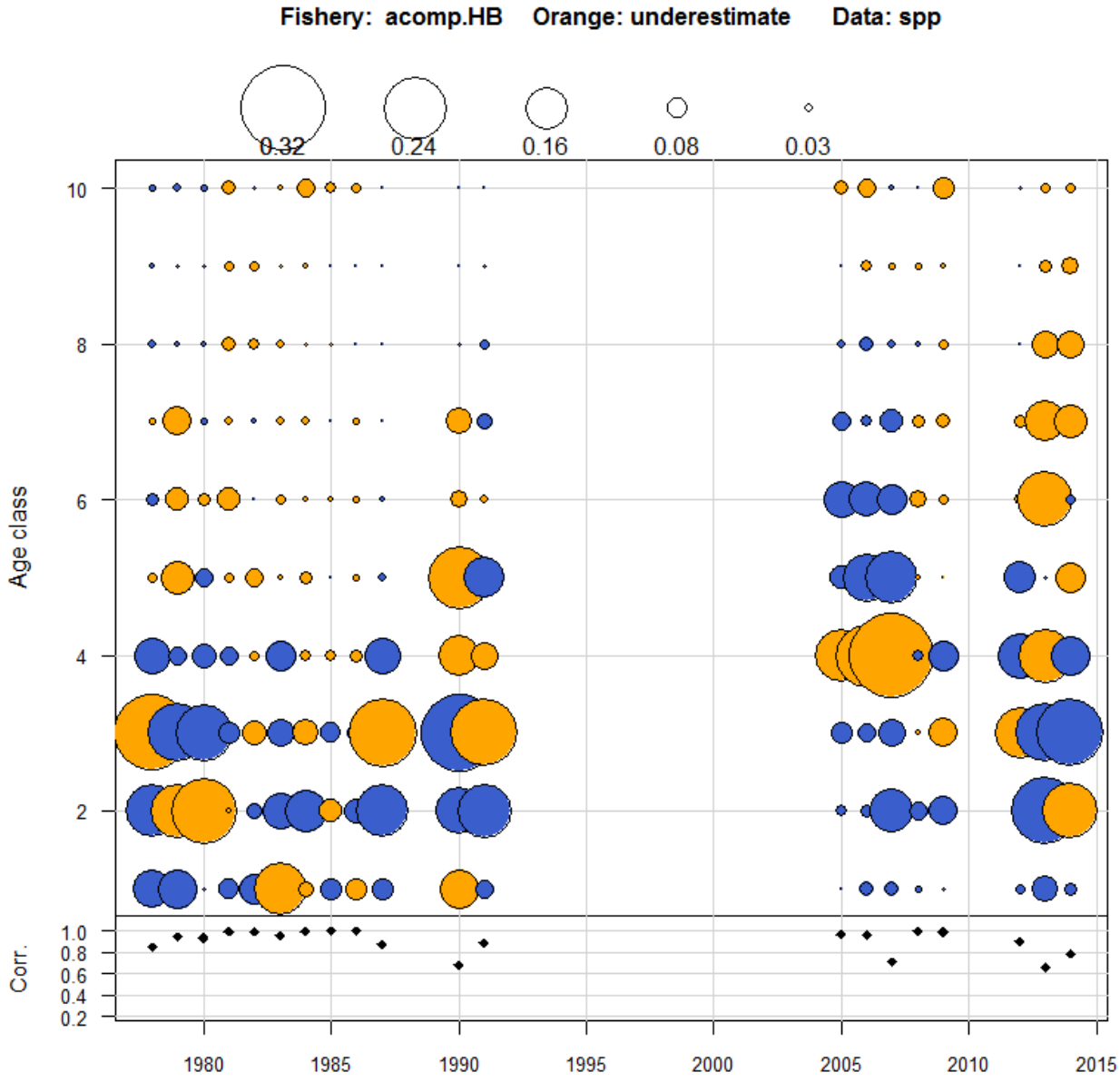

Proportion F at age Data: spp
Proportion of F
1.0
0.8
0.6
0.4
0.2
0.0
1950
1960
1970
1980
1990
2000
2010
Year
Age
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

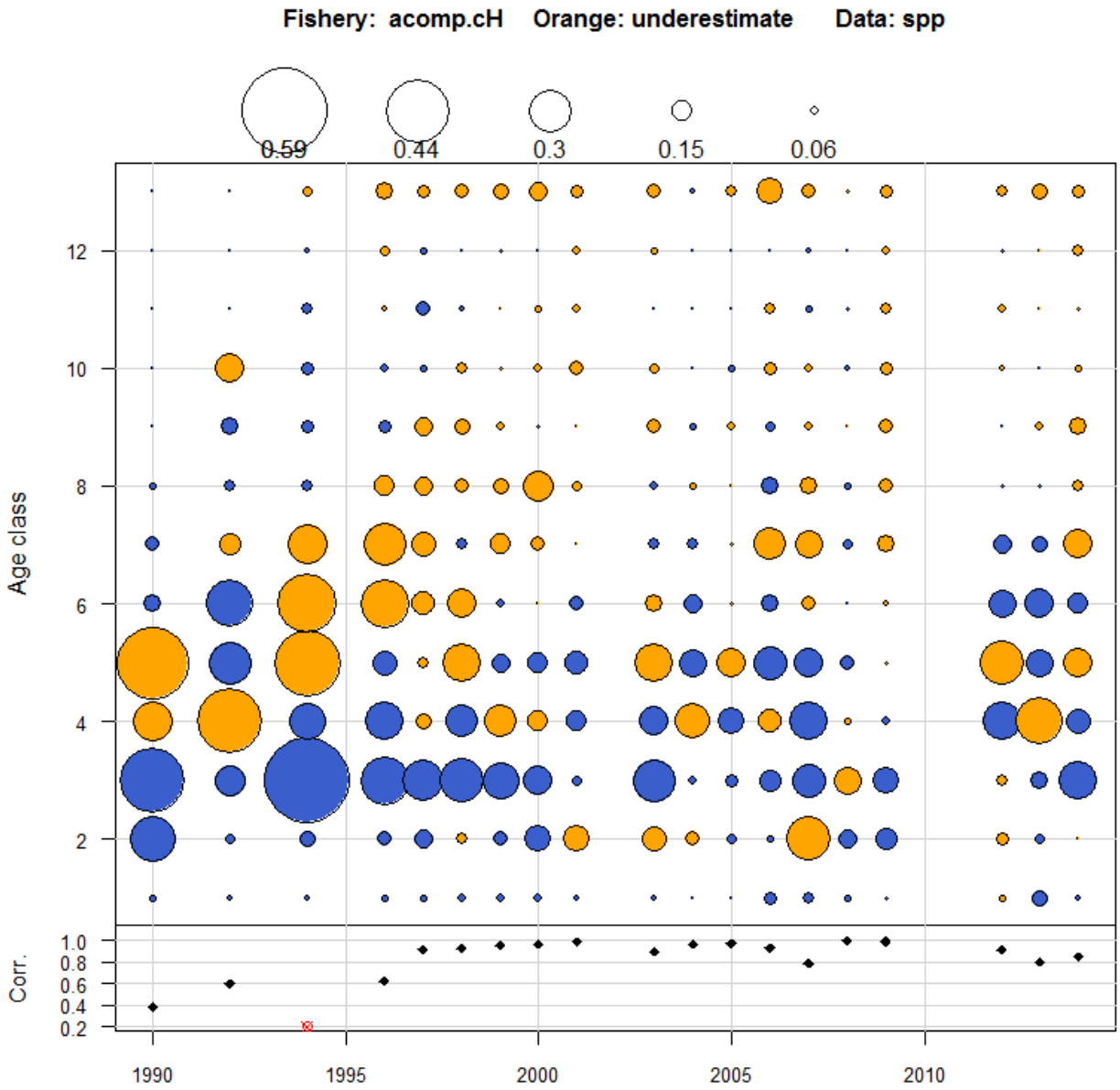

Proportion Z at age Data: spp
Proportion of Z
1.0
0.8
0.6
0.4
0.2
0.0
1950
1960
1970
1980
1990
2000
2010
Year
Age
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

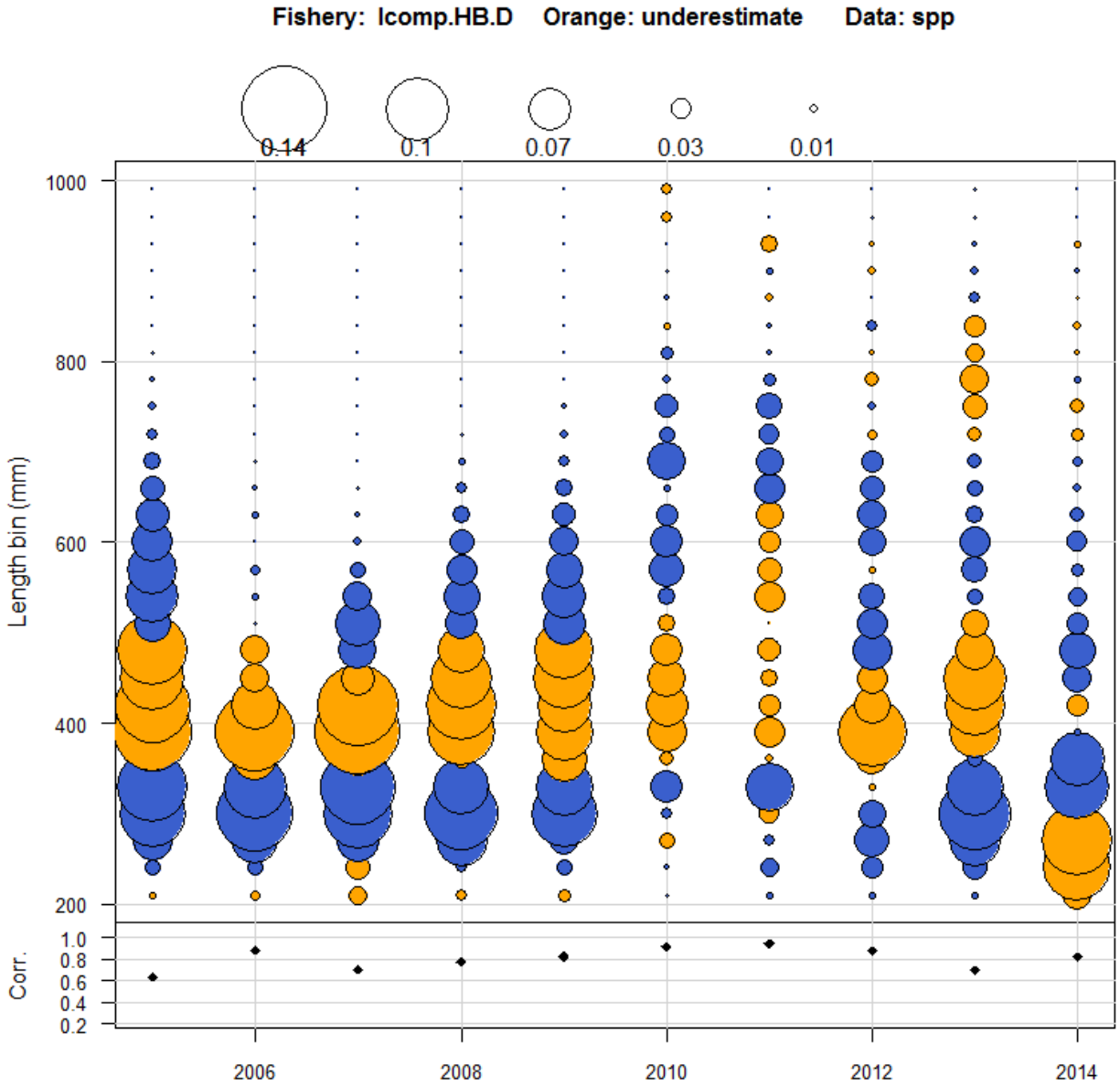

reprod Data: spp
reprod
Age
0.0
0.5
1.0
1.5
2.0
2.5
5
10
15
20

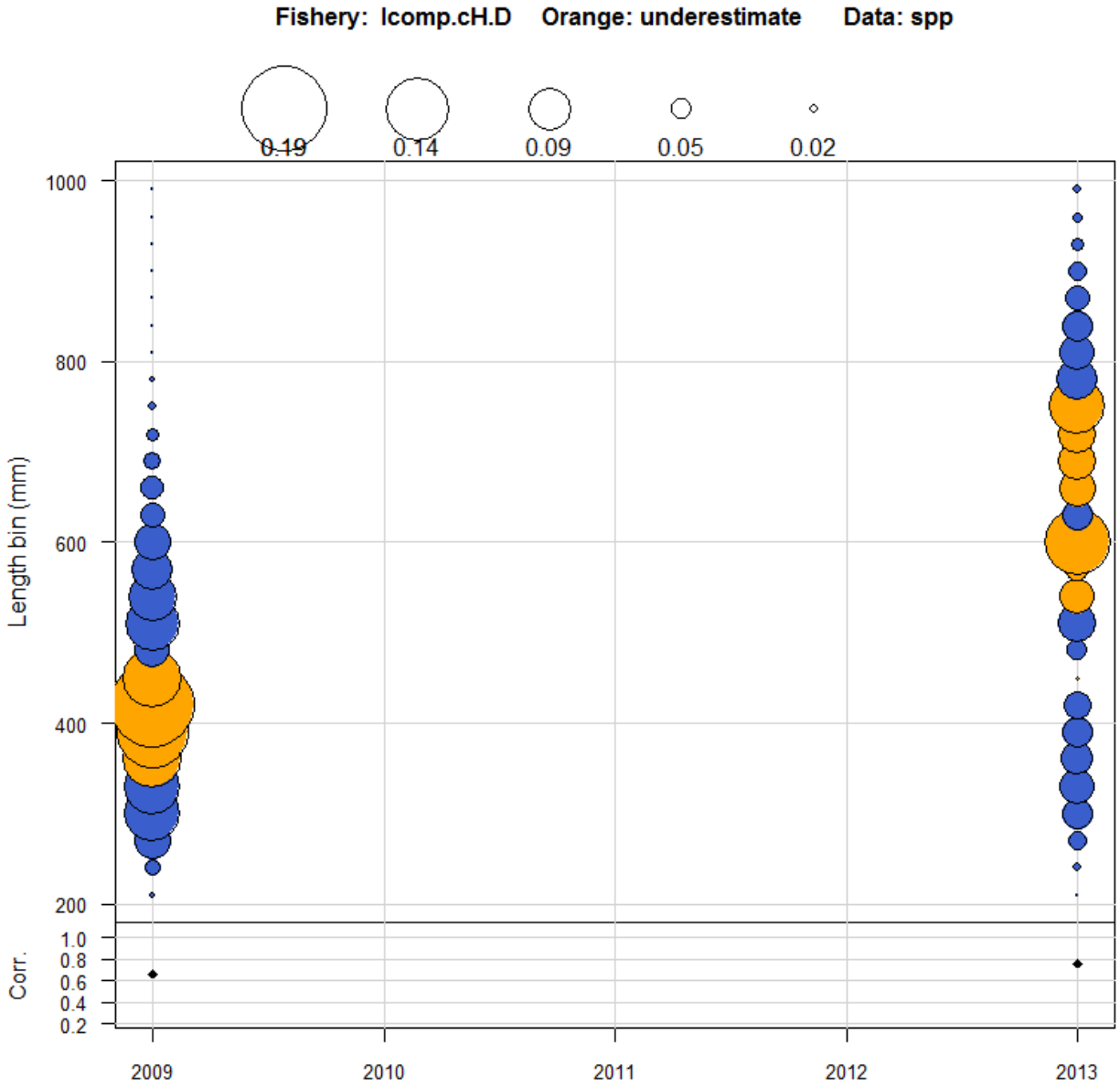

Fishery: acomp.GR Orange: underestimate Data: spp
0.22
0.16
0.11
0.05
0.02
Age class
12
10
8
6
4
2
Corr.
1.0
0.8
0.6
0.4
0.2
2002
2004
2006
2008
2010
2012
2014

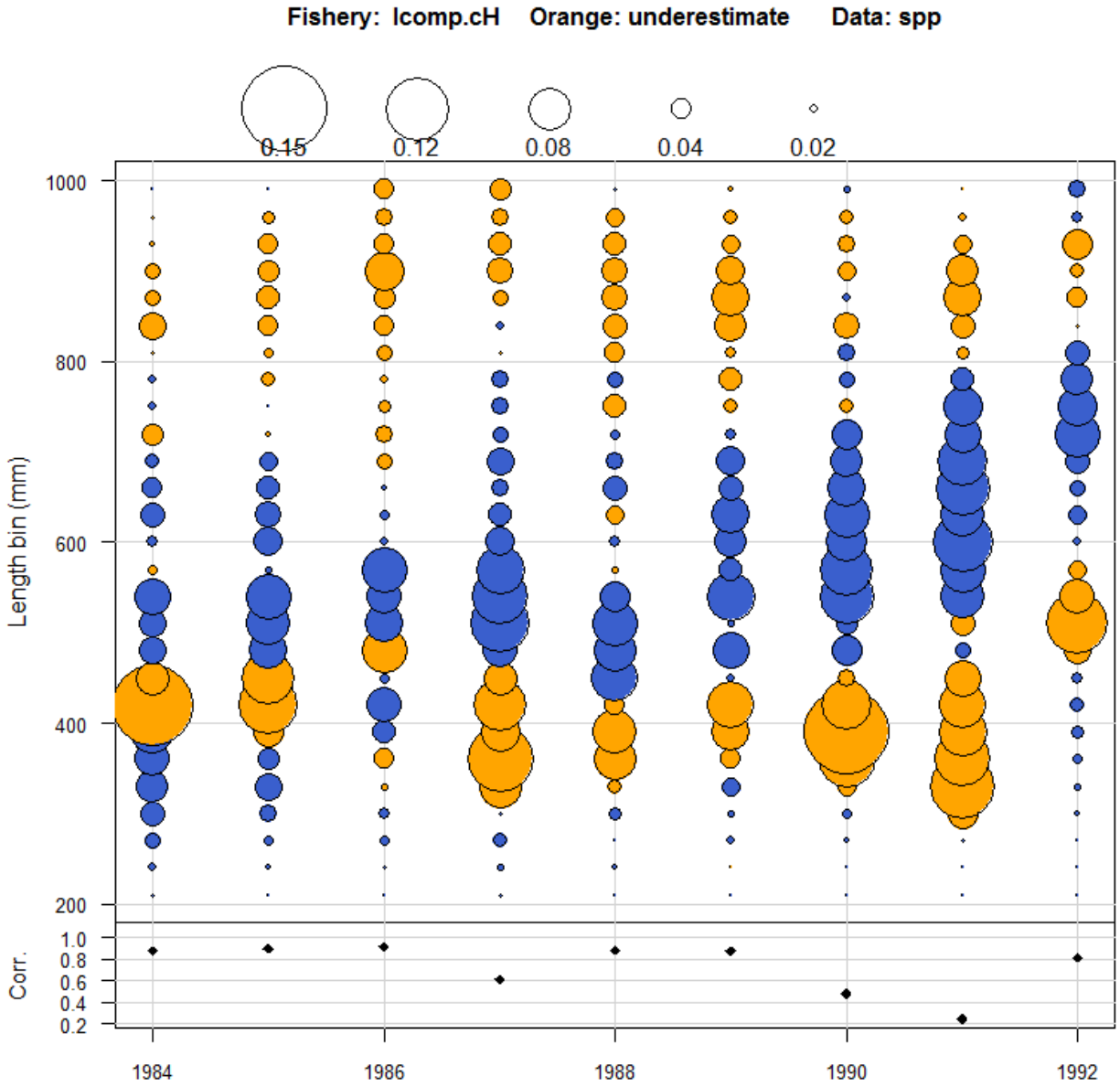

Fishery: acomp.CVT Orange: underestimate Data: spp
0.22
0.16
0.11
0.05
0.02
Age class
12
10
8
6
4
2
Corr.
1.0
0.8
0.6
0.4
0.2
2010
2011
2012
2013
2014

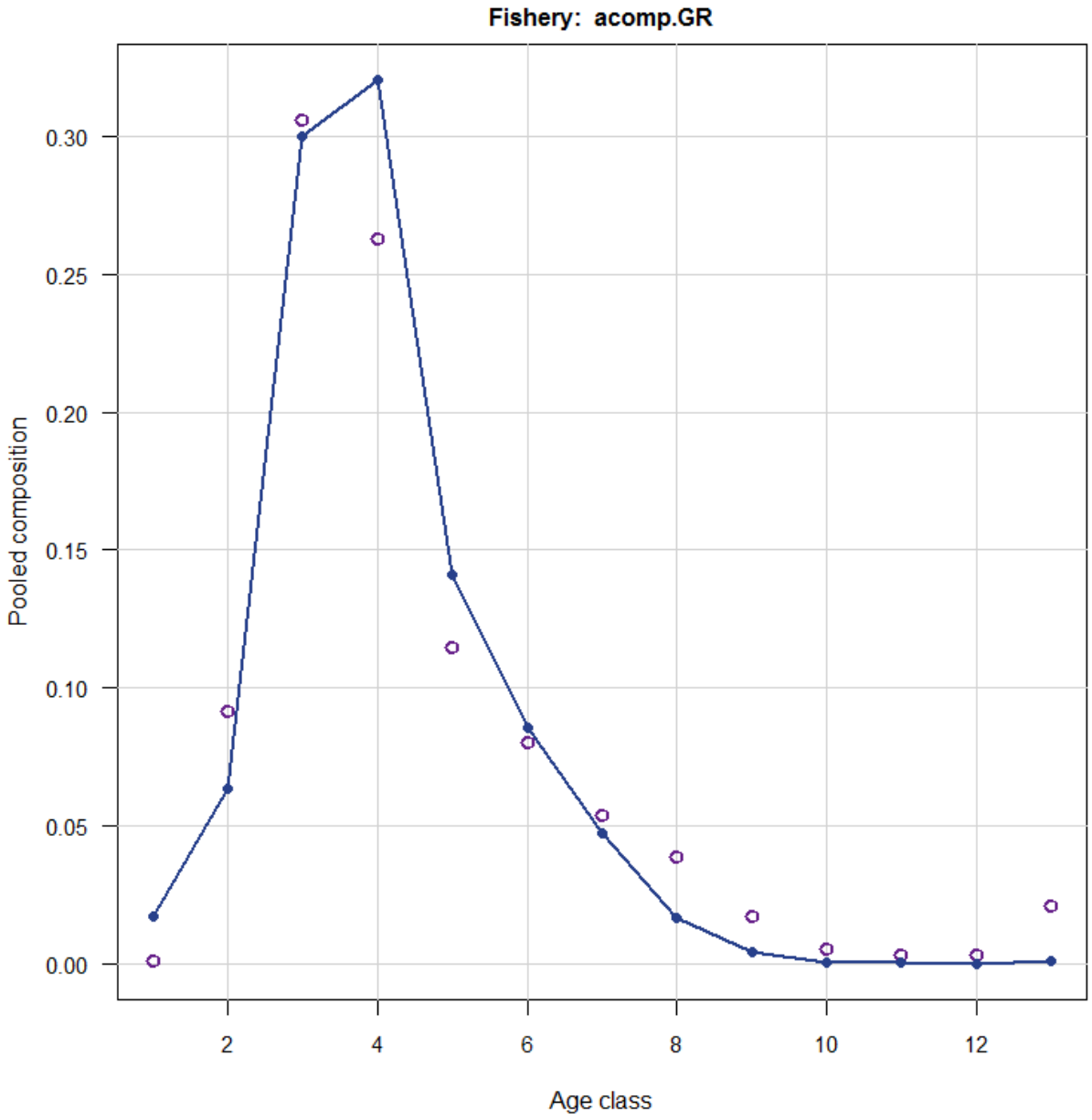

Fishery: acomp.HB Orange: underestimate Data: spp
0.32
0.24
0.16
0.08
0.03
Age class
10
8
6
4
2
Corr.
1.0
0.8
0.6
0.4
0.2
1980
1985
1990
1995
2000
2005
2010
2015

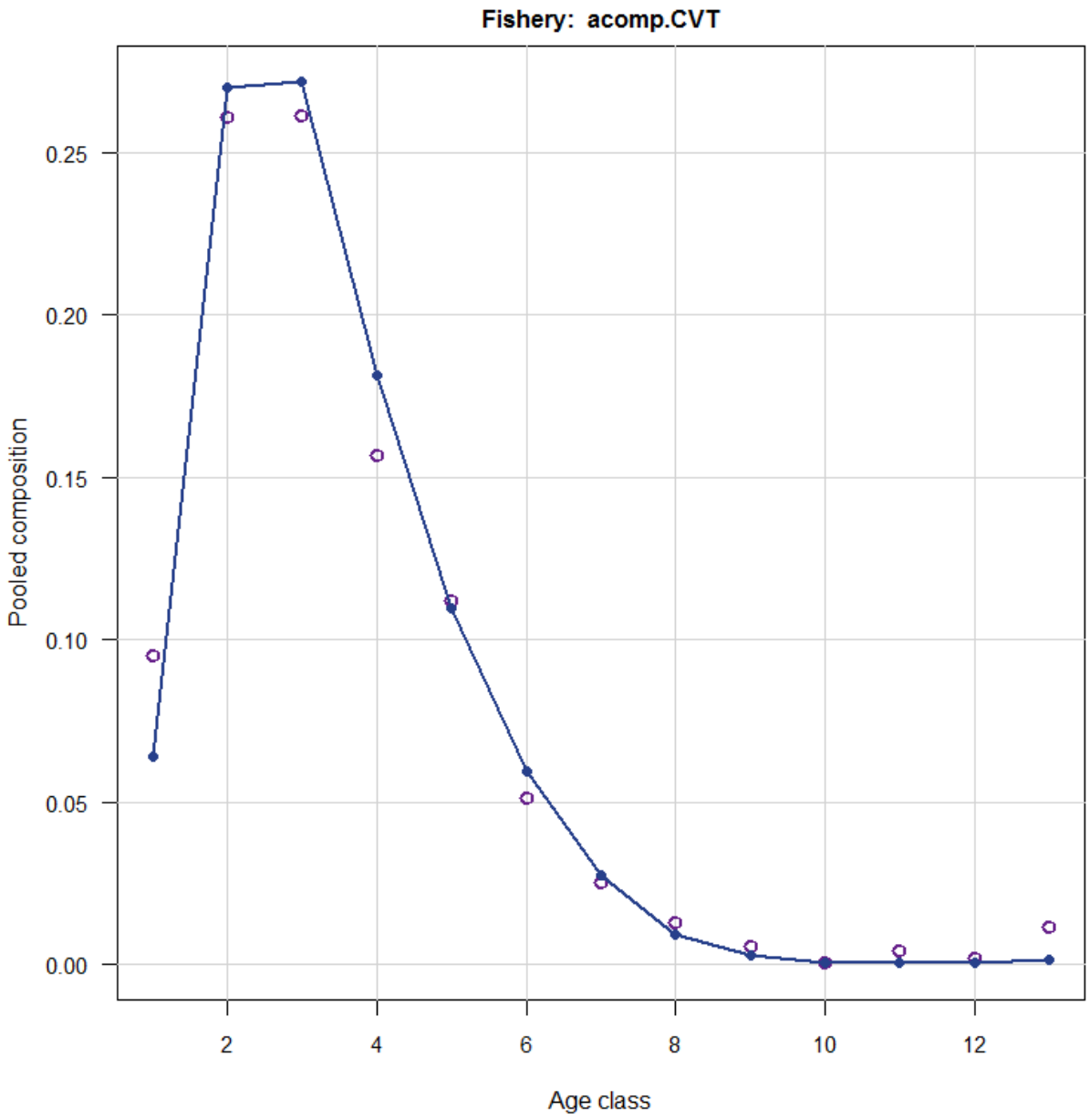

Fishery: acomp.cH Orange: underestimate Data: spp
0.59
0.44
0.3
0.15
0.06
Age class
12
10
8
6
4
2
Corr.
1.0
0.8
0.6
0.4
0.2
1990
1995
2000
2005
2010

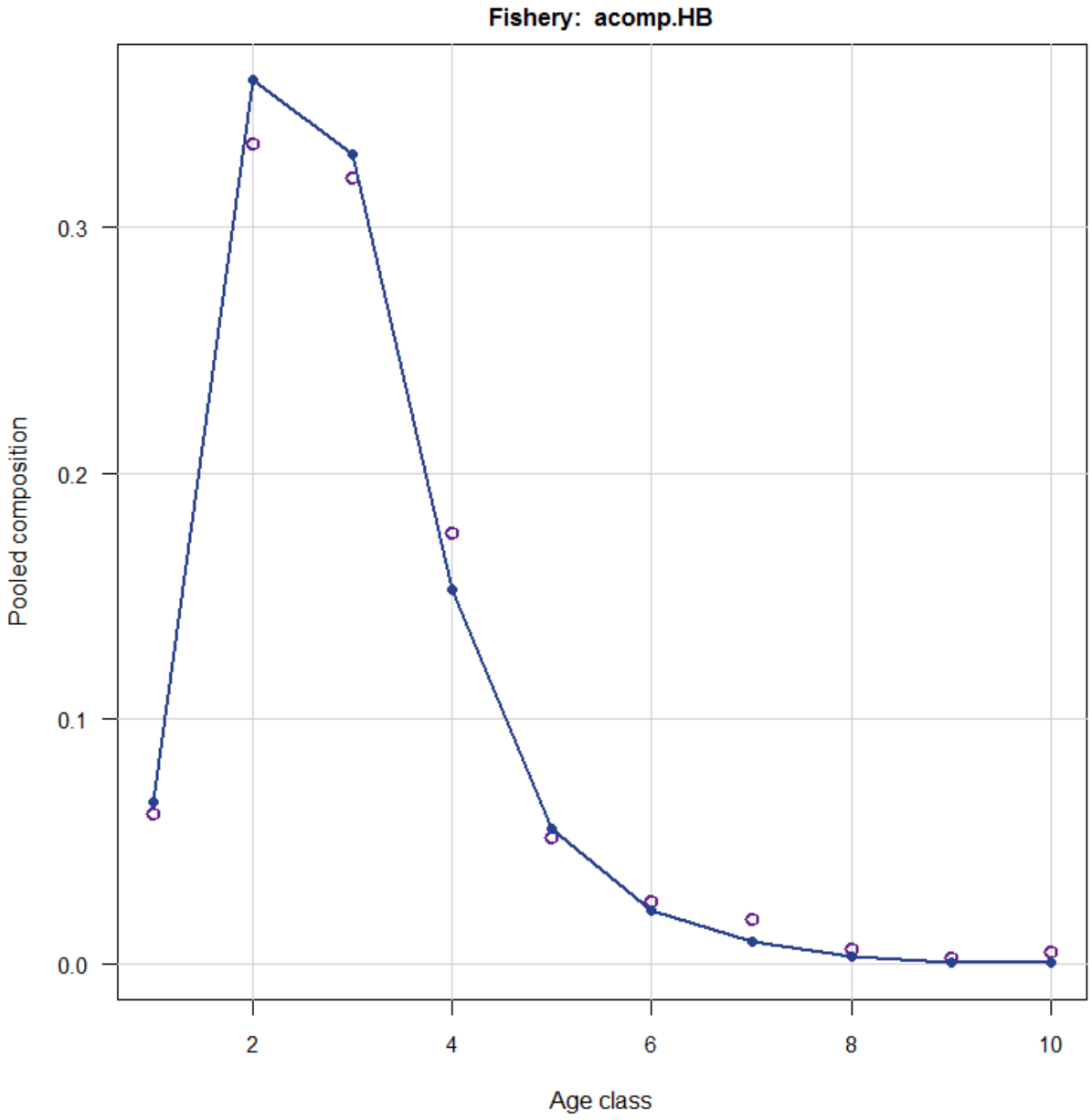

Fishery: lcomp.HB.D Orange: underestimate Data: spp
0.14
0.1
0.07
0.03
0.01
Length bin (mm)
1000
800
600
400
200
Corr.
1.0
0.8
0.6
0.4
0.2
2006
2008
2010
2012
2014

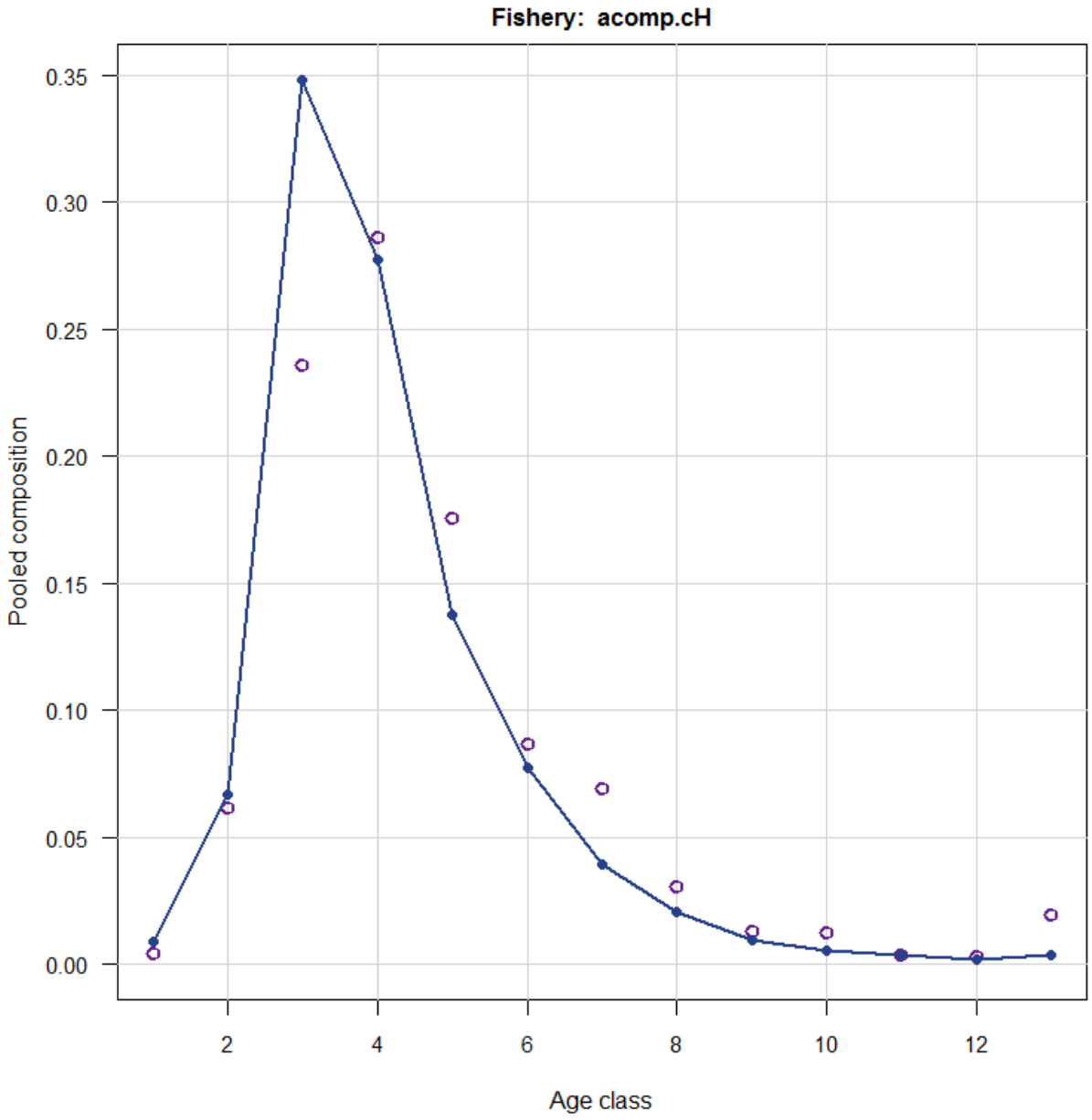

Fishery: lcomp.cH.D Orange: underestimate Data: spp
0.19
0.14
0.09
0.05
0.02
Length bin (mm)
1000
800
600
400
200
Corr.
1.0
0.8
0.6
0.4
0.2
2009
2010
2011
2012
2013

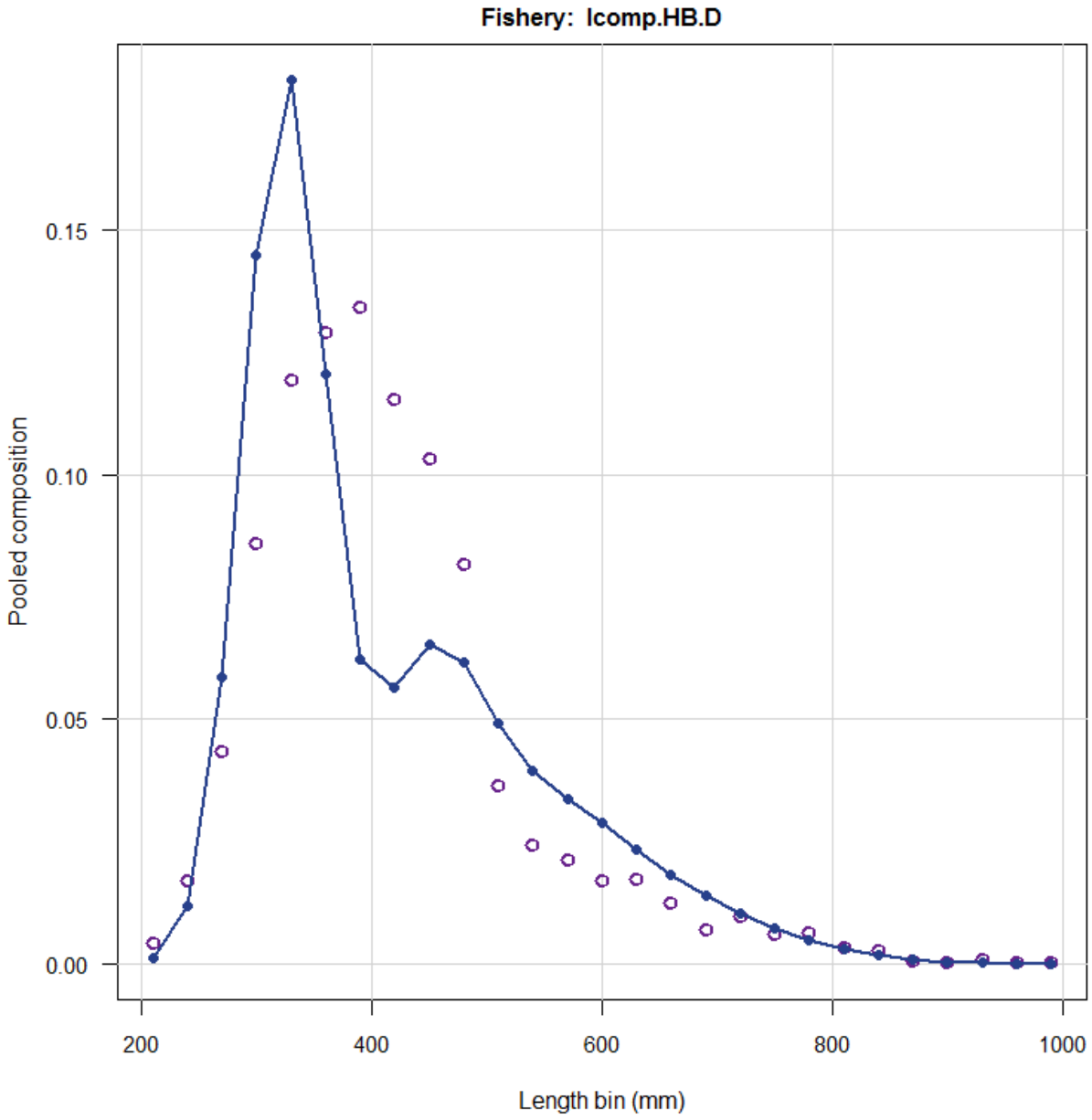

Fishery: lcomp.cH Orange: underestimate Data: spp
0.15
0.12
0.08
0.04
0.02
Length bin (mm)
1000
800
600
400
200
Corr.
1.0
0.8
0.6
0.4
0.2
1984
1986
1988
1990
1992

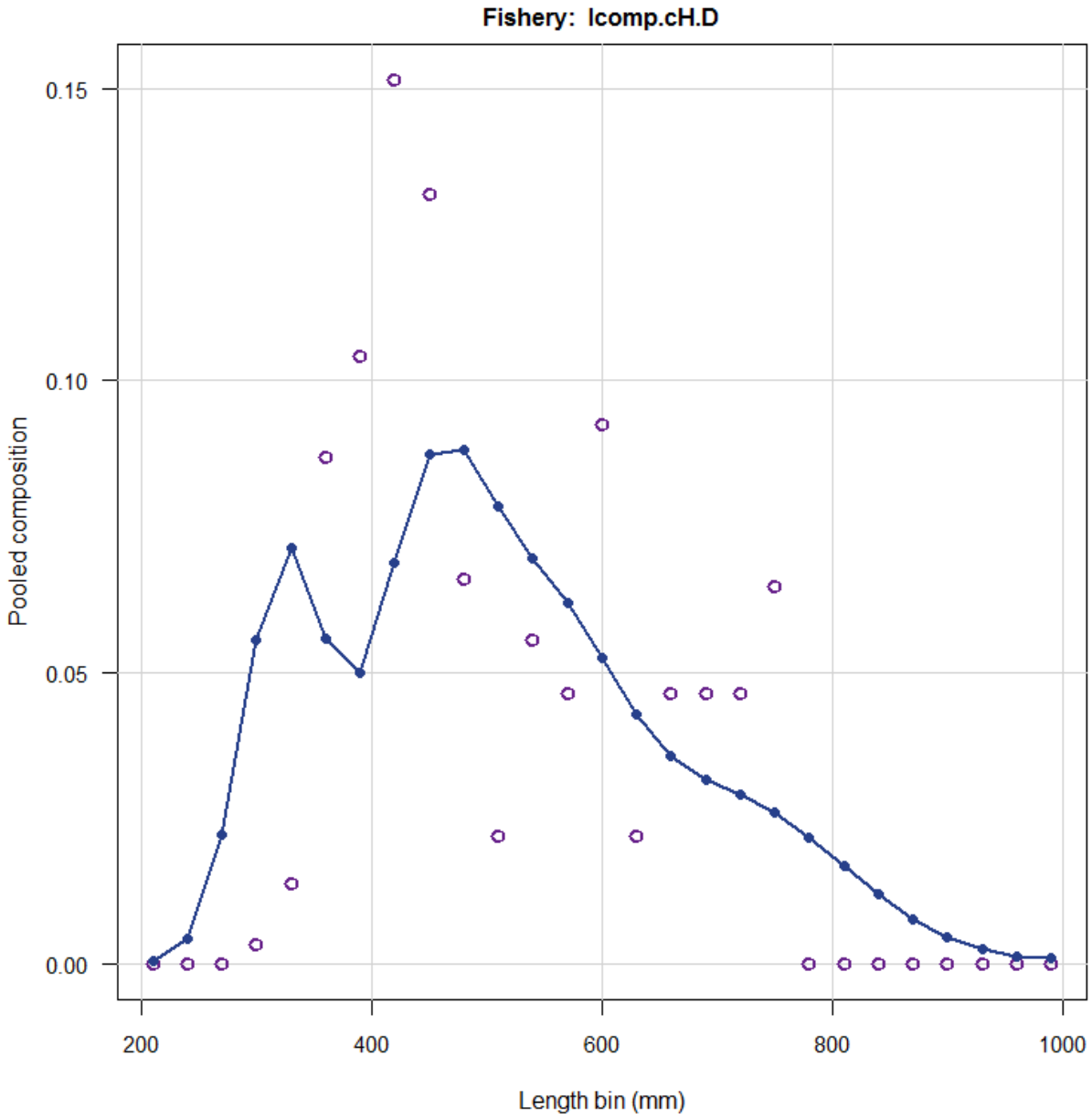

Fishery: acomp.GR
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
0.30
2
4
6
8
10
12
Age class

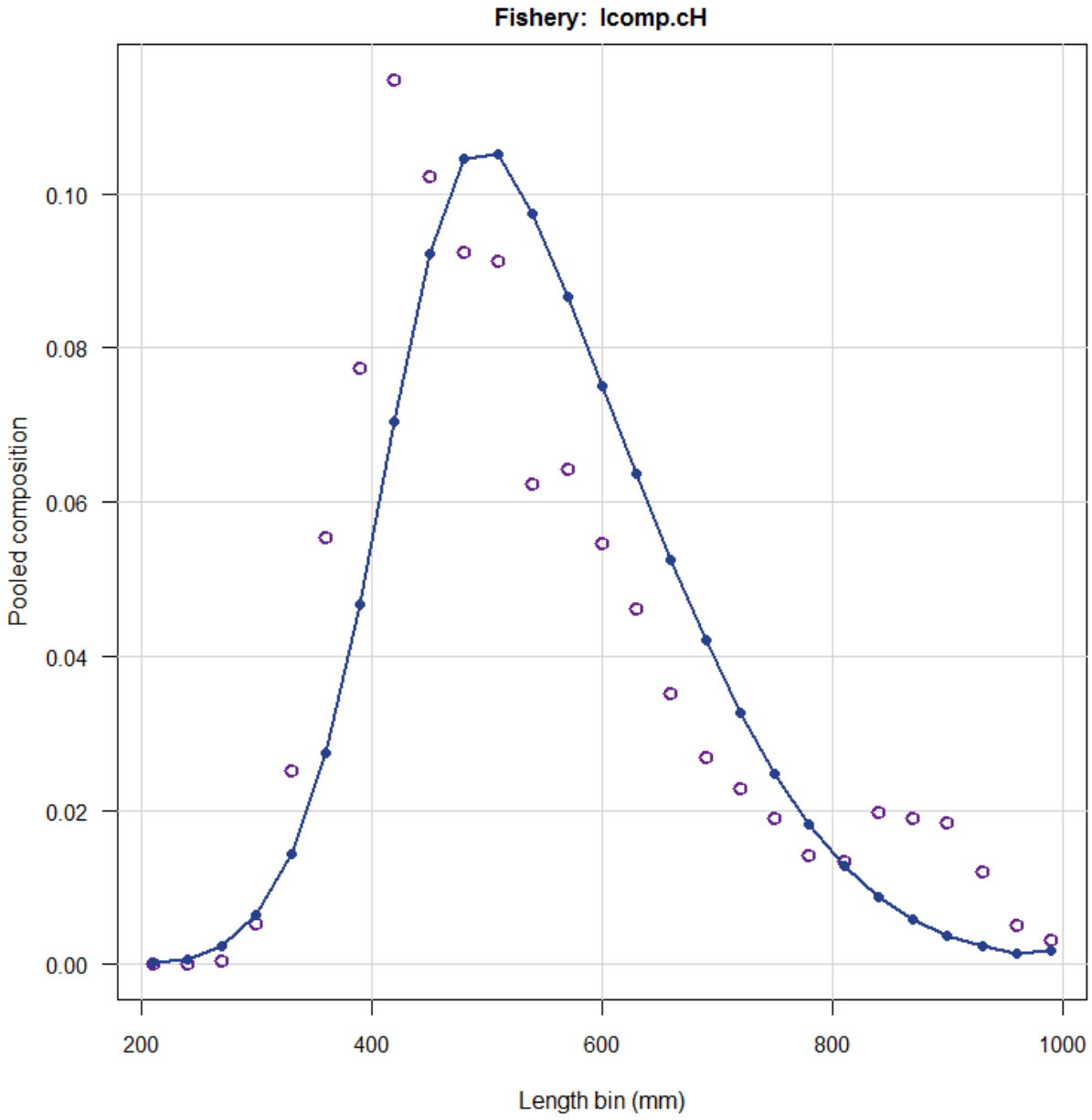

Fishery: acomp.CVT
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
2
4
6
8
10
12
Age class


Fishery: acomp.HB
Pooled composition
0.0
0.1
0.2
0.3
2
4
6
8
10
Age class


Fishery: acomp.cH
Pooled composition
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
2
4
6
8
10
12
Age class


Fishery: lcomp.HB.D
Pooled composition
0.00
0.05
0.10
0.15
200
400
600
800
1000
Length bin (mm)


Fishery: lcomp.cH.D
0.15
0.10
0.05
0.00
Pooled composition
200
400
600
800
1000
Length bin (mm)


Fishery: lcomp.cH
Pooled composition
0.00
0.02
0.04
0.06
0.08
0.10
200
400
600
800
1000
Length bin (mm)

ADMB tpl file for the BAM:

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

//##

//##  SEDAR 41  Red Snapper assessment November 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;
init_int endyr_selex_phase3;

//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;
init_int nages_agec_HB
//Vector of ages for age bins in age comps
init_vector agebins_agec(1,nages_agec);
init_vector agebins_agec_HB(1,nages_agec_HB);
```

```
//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;
 LOCAL_CALCS
                        n_iter_msy=n_iter_spr;
 END_CALCS

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

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

//################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 (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);

// Comm HL  Discards (1000 fish)
init_int styr_cH_D;
init_int endyr_cH_D;
```

```
init_int endyr_cH_D1;
init_vector obs_cH_released(styr_cH_D,endyr_cH_D);
init_vector cH_D_cv(styr_cH_D,endyr_cH_D);

//Comm HL discards length compositions
init_int nyr_cH_D_lenc;
init_ivector yrs_cH_D_lenc(1,nyr_cH_D_lenc);
init_vector nsamp_cH_D_lenc(1,nyr_cH_D_lenc);
init_vector nfish_cH_D_lenc(1,nyr_cH_D_lenc);
init_matrix obs_cH_D_lenc(1,nyr_cH_D_lenc,1,nlenbins);

//####################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 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);

// HB discards CPUE
init_int styr_HB_D_cpue;
init_int endyr_HB_D_cpue;
init_vector obs_HB_D_cpue(styr_HB_D_cpue,endyr_HB_D_cpue);//Observed CPUE
init_vector HB_D_cpue_cv(styr_HB_D_cpue,endyr_HB_D_cpue); //CV of cpue

// HB  Discards (1000 fish)
init_int styr_HB_D;
init_int endyr_HB_D;
init_int endyr_HB_D1;
init_vector obs_HB_released(styr_HB_D,endyr_HB_D);
init_vector HB_D_cv(styr_HB_D,endyr_HB_D);

//HB discards 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);

//####################Private/Charterboat Recreational fleet ##########################################
// 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
```

```
//  Discards (1000 fish)
init_int styr_GR_D;
init_int endyr_GR_D;
init_int endyr_GR_D1;
init_vector obs_GR_released(styr_GR_D,endyr_GR_D);
init_vector GR_D_cv(styr_GR_D,endyr_GR_D);

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

//###MARMAP CVT/VID combination#################  ***KIS
//CPUE
init_int styr_CVT_cpue;
init_int endyr_CVT_cpue;
init_vector obs_CVT_cpue(styr_CVT_cpue,endyr_CVT_cpue);   //Observed CPUE
init_vector CVT_cpue_cv(styr_CVT_cpue,endyr_CVT_cpue);    //CV of cpue

// Age Compositions
init_int nyr_CVT_agec;
init_ivector yrs_CVT_agec(1,nyr_CVT_agec);
init_vector nsamp_CVT_agec(1,nyr_CVT_agec);
init_vector nfish_CVT_agec(1,nyr_CVT_agec);
init_matrix obs_CVT_agec(1,nyr_CVT_agec,1,nages_agec);
```

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

//-- BAM DATA_SECTION: parameter section

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

//##################Single Parameter values and initial guesses #################################

// von Bert parms in TL mm population

init_vector set_Linf(1,7);

init_vector set_K(1,7);

init_vector set_t0(1,7);

init_vector set_len_cv(1,7);


//von Bert parms for all fisheries in periods 1 and 3

init_vector set_Linf_L(1,7);

init_vector set_K_L(1,7);

init_vector set_t0_L(1,7);

init_vector set_len_cv_L(1,7);


// von Bert parms in TL mm all fleets, 20in size limit

init_vector set_Linf_20(1,7);

init_vector set_K_20(1,7);

init_vector set_t0_20(1,7);

init_vector set_len_cv_20(1,7);


//Scalar used only for computing MSST. For gag, this might be 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

init_vector set_selpar_A50_cH1(1,7);
init_vector set_selpar_slope_cH1(1,7);

init_vector set_selpar_A50_cH2(1,7);
init_vector set_selpar_slope_cH2(1,7);

init_vector set_selpar_A50_cH3(1,7); //***KIS for the last selex timeblock
init_vector set_selpar_slope_cH3(1,7);

init_vector set_selpar_A50_HB1(1,7);
init_vector set_selpar_slope_HB1(1,7);
init_vector set_selpar_A502_HB1(1,7);
init_vector set_selpar_slope2_HB1(1,7);

init_vector set_selpar_A50_HB2(1,7);
init_vector set_selpar_slope_HB2(1,7);
init_vector set_selpar_A502_HB2(1,7);
init_vector set_selpar_slope2_HB2(1,7);

init_vector set_selpar_A50_HB3(1,7); //***KIS for the last selex timeblock
init_vector set_selpar_slope_HB3(1,7);
init_vector set_selpar_A502_HB3(1,7);
init_vector set_selpar_slope2_HB3(1,7);
```

```
init_vector set_selpar_A50_GR2(1,7); //***KIS for the last selex timeblock
init_vector set_selpar_slope_GR2(1,7);
init_vector set_selpar_A502_GR2(1,7);
init_vector set_selpar_slope2_GR2(1,7);

init_vector set_selpar_A50_GR3(1,7); //***KIS for the last selex timeblock
init_vector set_selpar_slope_GR3(1,7);
//init_vector set_selpar_A502_GR3(1,7);
//init_vector set_selpar_slope2_GR3(1,7);

init_vector set_selpar_A50_CVT(1,7); //***KIS for the CVT and VID indices
init_vector set_selpar_slope_CVT(1,7);

init_vector set_selpar_A50_cH2_D(1,7);  //***KIS for the cH disc 92-2009
init_vector set_selpar_slope_cH2_D(1,7);
init_vector set_selpar_A502_cH2_D(1,7);
init_vector set_selpar_slope2_cH2_D(1,7);

init_vector set_selpar_A50_cH3_D(1,7);  //***KIS for the cH disc 2010-2014
init_vector set_selpar_slope_cH3_D(1,7);

init_vector set_selpar_A50_HB2_D(1,7);  //***KIS for the HB disc 92-2009
init_vector set_selpar_slope_HB2_D(1,7);
init_vector set_selpar_A502_HB2_D(1,7);
init_vector set_selpar_slope2_HB2_D(1,7);

init_vector set_selpar_A50_HB3_D(1,7);  //***KIS for the HB disc 2010-2014
init_vector set_selpar_slope_HB3_D(1,7);
init_vector set_selpar_A502_HB3_D(1,7);
```

```
init_vector set_selpar_slope2_HB3_D(1,7);

//--index catchability----------------------------------------------------------------------------
init_vector set_log_q_cH(1,7);      //catchability coefficient (log) for comm handline index
init_vector set_log_q_HB(1,7);      //catchability coefficient (log) for headboat index
init_vector set_log_q_HB_D(1,7);      //catchability coefficient (log) for headboat discard index
init_vector set_log_q_CVT(1,7);      //catchability coefficient (log) for SERFS index  ***KIS

//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_cH_D(1,7);
init_vector set_log_avg_F_HB_D(1,7);
init_vector set_log_avg_F_GR_D(1,7);

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

```
init_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_cH_D_vals(styr_cH_D,endyr_cH_D);

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;

init_number set_w_D;            //weight for discards

init_number set_w_I_cH;         //weight for comm handline index

init_number set_w_I_HB;         //weight for headboat index

init_number set_w_I_HB_D;         //weight for MRFSS index

init_number set_w_I_CVT;

init_number set_w_lc_cH;        //weight for comm handline len comps

init_number set_w_lc_cH_D;

//init_number set_w_lc_HB;        //weight for headboat len comps

init_number set_w_lc_HB_D;                      //weight for the headboat discard length comps  ***KIS***

//init_number set_w_lc_CVT;                     //weight for the SERFS length comps  ***KIS***

//init_number set_w_lc_GR;                      //weight for the recreational length comps  ***KIS***

init_number set_w_ac_cH;        //weight for comm handline age comps

init_number set_w_ac_HB;        //weight for headboat age comps

init_number set_w_ac_CVT;                       //weight for the SERFS age comps  ***KIS***
```

```
init_number set_w_ac_GR;                //weight for the Recreational age comps  ***KIS***

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

init_number set_w_rec;          //for fitting S-R curve

init_number set_w_rec_early;    //additional constraint on early years recruitment

init_number set_w_rec_end;      //additional constraint on ending years recruitment

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

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

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

//-- BAM DATA_SECTION: miscellaneous stuff section

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


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

init_number wgtpar_a;

init_number wgtpar_b;


//weight(whole weight)-gonad weight (units=g) relationship: GW=a+b*W

init_number ww2gw;


//Maturity and proportion female at age

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


//Fecundity-length relationship

init_vector prop_f_obs(1,nages);

init_vector set_fecpar_batches(1,nages);

init_number set_fecpar_a;

init_number set_fecpar_b;

init_number fecpar_scale;

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_cH1;
init_number set_Dmort_HB1;
init_number set_Dmort_GR1;
init_number set_Dmort_cH2;
init_number set_Dmort_HB2;
init_number set_Dmort_GR2;

//scalar multiple applied to observed historic recreational landings in MCB analysis
init_number historic_Lrec_scale;

//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_cH_D_lenc;
//init_number minSS_HB_lenc;
init_number minSS_HB_D_lenc;
//init_number minSS_CVT_lenc;
//init_number minSS_GR_lenc;

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


PARAMETER_SECTION

 LOCAL_CALCS
  const double Linf_LO=set_Linf(2); const double Linf_HI=set_Linf(3); const double Linf_PH=set_Linf(4);
  const double K_LO=set_K(2); const double K_HI=set_K(3); const double K_PH=set_K(4);
  const double t0_LO=set_t0(2); const double t0_HI=set_t0(3); const double t0_PH=set_t0(4);
```

```
  const double len_cv_LO=set_len_cv(2); const double len_cv_HI=set_len_cv(3); const double
len_cv_PH=set_len_cv(4);


  const double 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);


  const double Linf_20_LO=set_Linf_20(2); const double Linf_20_HI=set_Linf_20(3); const double
Linf_20_PH=set_Linf_20(4);

  const double K_20_LO=set_K_20(2); const double K_20_HI=set_K_20(3); const double K_20_PH=set_K_20(4);

  const double t0_20_LO=set_t0_20(2); const double t0_20_HI=set_t0_20(3); const double t0_20_PH=set_t0_20(4);

  const double len_cv_20_LO=set_len_cv_20(2); const double len_cv_20_HI=set_len_cv_20(3); const double
len_cv_20_PH=set_len_cv_20(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_A50_cH1_LO=set_selpar_A50_cH1(2); const double
selpar_A50_cH1_HI=set_selpar_A50_cH1(3); const double selpar_A50_cH1_PH=set_selpar_A50_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_A50_cH2_LO=set_selpar_A50_cH2(2); const double
selpar_A50_cH2_HI=set_selpar_A50_cH2(3); const double selpar_A50_cH2_PH=set_selpar_A50_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_A50_cH3_LO=set_selpar_A50_cH3(2); const double
selpar_A50_cH3_HI=set_selpar_A50_cH3(3); const double selpar_A50_cH3_PH=set_selpar_A50_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_A50_HB1_LO=set_selpar_A50_HB1(2); const double
selpar_A50_HB1_HI=set_selpar_A50_HB1(3); const double selpar_A50_HB1_PH=set_selpar_A50_HB1(4);

  const double selpar_slope_HB1_LO=set_selpar_slope_HB1(2); const double
selpar_slope_HB1_HI=set_selpar_slope_HB1(3); const double selpar_slope_HB1_PH=set_selpar_slope_HB1(4);

  const double selpar_A502_HB1_LO=set_selpar_A502_HB1(2); const double
selpar_A502_HB1_HI=set_selpar_A502_HB1(3); const double selpar_A502_HB1_PH=set_selpar_A502_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_A50_HB2_LO=set_selpar_A50_HB2(2); const double
selpar_A50_HB2_HI=set_selpar_A50_HB2(3); const double selpar_A50_HB2_PH=set_selpar_A50_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_A502_HB2_LO=set_selpar_A502_HB2(2); const double
selpar_A502_HB2_HI=set_selpar_A502_HB2(3); const double selpar_A502_HB2_PH=set_selpar_A502_HB2(4);

  const double selpar_slope2_HB2_LO=set_selpar_slope2_HB2(2); const double
selpar_slope2_HB2_HI=set_selpar_slope2_HB2(3); const double
selpar_slope2_HB2_PH=set_selpar_slope2_HB2(4);

  const double selpar_A50_HB3_LO=set_selpar_A50_HB3(2); const double
selpar_A50_HB3_HI=set_selpar_A50_HB3(3); const double selpar_A50_HB3_PH=set_selpar_A50_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_A502_HB3_LO=set_selpar_A502_HB3(2); const double
selpar_A502_HB3_HI=set_selpar_A502_HB3(3); const double selpar_A502_HB3_PH=set_selpar_A502_HB3(4);

  const double selpar_slope2_HB3_LO=set_selpar_slope2_HB3(2); const double
selpar_slope2_HB3_HI=set_selpar_slope2_HB3(3); const double
selpar_slope2_HB3_PH=set_selpar_slope2_HB3(4);



  const double selpar_A50_GR2_LO=set_selpar_A50_GR2(2); const double
selpar_A50_GR2_HI=set_selpar_A50_GR2(3); const double selpar_A50_GR2_PH=set_selpar_A50_GR2(4);

  const double selpar_slope_GR2_LO=set_selpar_slope_GR2(2); const double
selpar_slope_GR2_HI=set_selpar_slope_GR2(3); const double selpar_slope_GR2_PH=set_selpar_slope_GR2(4);
```

```
  const double selpar_A502_GR2_LO=set_selpar_A502_GR2(2); const double
selpar_A502_GR2_HI=set_selpar_A502_GR2(3); const double selpar_A502_GR2_PH=set_selpar_A502_GR2(4);

  const double selpar_slope2_GR2_LO=set_selpar_slope2_GR2(2); const double
selpar_slope2_GR2_HI=set_selpar_slope2_GR2(3); const double
selpar_slope2_GR2_PH=set_selpar_slope2_GR2(4);



  const double selpar_A50_GR3_LO=set_selpar_A50_GR3(2); const double
selpar_A50_GR3_HI=set_selpar_A50_GR3(3); const double selpar_A50_GR3_PH=set_selpar_A50_GR3(4);

  const double selpar_slope_GR3_LO=set_selpar_slope_GR3(2); const double
selpar_slope_GR3_HI=set_selpar_slope_GR3(3); const double selpar_slope_GR3_PH=set_selpar_slope_GR3(4);

  //const double selpar_A502_GR3_LO=set_selpar_A502_GR3(2); const double
selpar_A502_GR3_HI=set_selpar_A502_GR3(3); const double selpar_A502_GR3_PH=set_selpar_A502_GR3(4);

  //const double selpar_slope2_GR3_LO=set_selpar_slope2_GR3(2); const double
selpar_slope2_GR3_HI=set_selpar_slope2_GR3(3); const double
selpar_slope2_GR3_PH=set_selpar_slope2_GR3(4);



  const double selpar_A50_CVT_LO=set_selpar_A50_CVT(2); const double
selpar_A50_CVT_HI=set_selpar_A50_CVT(3); const double selpar_A50_CVT_PH=set_selpar_A50_CVT(4);

  const double selpar_slope_CVT_LO=set_selpar_slope_CVT(2); const double
selpar_slope_CVT_HI=set_selpar_slope_CVT(3); const double selpar_slope_CVT_PH=set_selpar_slope_CVT(4);



  const double selpar_A50_HB2_D_LO=set_selpar_A50_HB2_D(2); const double
selpar_A50_HB2_D_HI=set_selpar_A50_HB2_D(3); const double
selpar_A50_HB2_D_PH=set_selpar_A50_HB2_D(4);

  const double selpar_slope_HB2_D_LO=set_selpar_slope_HB2_D(2); const double
selpar_slope_HB2_D_HI=set_selpar_slope_HB2_D(3); const double
selpar_slope_HB2_D_PH=set_selpar_slope_HB2_D(4);

  const double selpar_A502_HB2_D_LO=set_selpar_A502_HB2_D(2); const double
selpar_A502_HB2_D_HI=set_selpar_A502_HB2_D(3); const double
selpar_A502_HB2_D_PH=set_selpar_A502_HB2_D(4);

  const double selpar_slope2_HB2_D_LO=set_selpar_slope2_HB2_D(2); const double
selpar_slope2_HB2_D_HI=set_selpar_slope2_HB2_D(3); const double
selpar_slope2_HB2_D_PH=set_selpar_slope2_HB2_D(4);



  const double selpar_A50_HB3_D_LO=set_selpar_A50_HB3_D(2); const double
selpar_A50_HB3_D_HI=set_selpar_A50_HB3_D(3); const double
selpar_A50_HB3_D_PH=set_selpar_A50_HB3_D(4);
```

```
  const double selpar_slope_HB3_D_LO=set_selpar_slope_HB3_D(2); const double
selpar_slope_HB3_D_HI=set_selpar_slope_HB3_D(3); const double
selpar_slope_HB3_D_PH=set_selpar_slope_HB3_D(4);

  const double selpar_A502_HB3_D_LO=set_selpar_A502_HB3_D(2); const double
selpar_A502_HB3_D_HI=set_selpar_A502_HB3_D(3); const double
selpar_A502_HB3_D_PH=set_selpar_A502_HB3_D(4);

  const double selpar_slope2_HB3_D_LO=set_selpar_slope2_HB3_D(2); const double
selpar_slope2_HB3_D_HI=set_selpar_slope2_HB3_D(3); const double
selpar_slope2_HB3_D_PH=set_selpar_slope2_HB3_D(4);



  const double selpar_A50_cH2_D_LO=set_selpar_A50_cH2_D(2); const double
selpar_A50_cH2_D_HI=set_selpar_A50_cH2_D(3); const double
selpar_A50_cH2_D_PH=set_selpar_A50_cH2_D(4);

  const double selpar_slope_cH2_D_LO=set_selpar_slope_cH2_D(2); const double
selpar_slope_cH2_D_HI=set_selpar_slope_cH2_D(3); const double
selpar_slope_cH2_D_PH=set_selpar_slope_cH2_D(4);

  const double selpar_A502_cH2_D_LO=set_selpar_A502_cH2_D(2); const double
selpar_A502_cH2_D_HI=set_selpar_A502_cH2_D(3); const double
selpar_A502_cH2_D_PH=set_selpar_A502_cH2_D(4);

  const double selpar_slope2_cH2_D_LO=set_selpar_slope2_cH2_D(2); const double
selpar_slope2_cH2_D_HI=set_selpar_slope2_cH2_D(3); const double
selpar_slope2_cH2_D_PH=set_selpar_slope2_cH2_D(4);



  const double selpar_A50_cH3_D_LO=set_selpar_A50_cH3_D(2); const double
selpar_A50_cH3_D_HI=set_selpar_A50_cH3_D(3); const double
selpar_A50_cH3_D_PH=set_selpar_A50_cH3_D(4);

  const double selpar_slope_cH3_D_LO=set_selpar_slope_cH3_D(2); const double
selpar_slope_cH3_D_HI=set_selpar_slope_cH3_D(3); const double
selpar_slope_cH3_D_PH=set_selpar_slope_cH3_D(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_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_HB_D_LO=set_log_q_HB_D(2); const double log_q_HB_D_HI=set_log_q_HB_D(3); const
double log_q_HB_D_PH=set_log_q_HB_D(4);

  const double log_q_CVT_LO=set_log_q_CVT(2); const double log_q_CVT_HI=set_log_q_CVT(3); const double
log_q_CVT_PH=set_log_q_CVT(4);
```

```
  const double 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_cH_D_LO=set_log_avg_F_cH_D(2); const double
log_avg_F_cH_D_HI=set_log_avg_F_cH_D(3); const double log_avg_F_cH_D_PH=set_log_avg_F_cH_D(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_cH_D_LO=set_log_F_dev_cH_D(1); const double
log_F_dev_cH_D_HI=set_log_F_dev_cH_D(2); const double log_F_dev_cH_D_PH=set_log_F_dev_cH_D(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 parms and conversions
  init_bounded_number Linf(Linf_LO,Linf_HI,Linf_PH);
  init_bounded_number K(K_LO,K_HI,K_PH);
  init_bounded_number t0(t0_LO,t0_HI,t0_PH);
  init_bounded_number len_cv_val(len_cv_LO,len_cv_HI,len_cv_PH);
  vector Linf_out(1,8);
  vector K_out(1,8);
  vector t0_out(1,8);
  vector len_cv_val_out(1,8);

  vector meanlen_TL(1,nages);   //mean total length (mm) at age all fish
  vector fecundity(1,nages);         // annual fec at age per mat female: fecpar_batches*(fecpar_a*FL^fecpar_b)
  vector fecpar_batches(1,nages);         //number of batches by age
  number fecpar_a;
  number fecpar_b;

  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

  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);
  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 meanlen_TL_L(1,nages);   //mean total 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
 vector wgt_klb_gut_L(1,nages);  //gutted wgt in 1000 lb
 vector wgt_lb_gut_L(1,nages);   //gutted wgt in lb

 init_bounded_number Linf_20(Linf_20_LO,Linf_20_HI,Linf_20_PH);
 init_bounded_number K_20(K_20_LO,K_20_HI,K_20_PH);
 init_bounded_number t0_20(t0_20_LO,t0_20_HI,t0_20_PH);
 init_bounded_number len_cv_val_20(len_cv_20_LO,len_cv_20_HI,len_cv_20_PH);
 vector Linf_20_out(1,8);
 vector K_20_out(1,8);
 vector t0_20_out(1,8);
 vector len_cv_val_20_out(1,8);
 vector meanlen_TL_20(1,nages);   //mean total length (mm) at age all fish

 vector wgt_g_20(1,nages);        //whole wgt in g
 vector wgt_kg_20(1,nages);       //whole wgt in kg
 vector wgt_mt_20(1,nages);       //whole wgt in mt
 vector wgt_klb_20(1,nages);      //whole wgt in 1000 lb
 vector wgt_lb_20(1,nages);       //whole wgt in lb
```

```
  matrix len_cH_mm(styr,endyr,1,nages);          //mean length at age of commercial handline landings in mm (may differ from popn mean)

  matrix wholewgt_cH_klb(styr,endyr,1,nages);    //whole wgt of commercial handline landings in 1000 lb

  matrix len_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_cH_D_mm(styr,endyr,1,nages);          //mean length at age of commercial handline discards in mm (may differ from popn mean)

  matrix wholewgt_cH_D_klb(styr,endyr,1,nages);    //whole wgt of commercial handline discards 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

  number zscore_len;                          //standardized normal values used for computing lenprob

  vector cprob_lenvec(1,nlenbins);            //cumulative probabilities used for computing lenprob

  number zscore_lzero;                        //standardized normal values for length = 0

  number cprob_lzero;                         //length probability mass below zero, used for computing lenprob


  number zscore_len_L;                        //standardized normal values used for computing lenprob

  vector cprob_lenvec_L(1,nlenbins);          //cumulative probabilities used for computing lenprob

  number zscore_lzero_L;                      //standardized normal values for length = 0

  number cprob_lzero_L;                       //length probability mass below zero, used for computing lenprob


  number zscore_len_20;                       //standardized normal values used for computing lenprob
```

```
  vector cprob_lenvec_20(1,nlenbins);            //cumulative probabilities used for computing lenprob
  number zscore_lzero_20;                        //standardized normal values for length = 0
  number cprob_lzero_20;                         //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_cH_D(1,nages,1,nlenbins);     //distn of size at age in cH discards  ***KIS***
  matrix lenprob_HB(1,nages,1,nlenbins);     //distn of size at age in HB (rec)
  matrix lenprob_HB_D(1,nages,1,nlenbins);     //distn of size at age in HB discards  ***KIS***
  matrix lenprob_CVT(1,nages,1,nlenbins);     //distn of size at age in CVT  ***KIS***
  matrix lenprob_GR(1,nages,1,nlenbins);     //distn of size at age in GR  ***KIS***
  matrix lenprob_20(1,nages,1,nlenbins);     //distn of size at age under 20" size limit  ***KIS***
  matrix lenprob_L(1,nages,1,nlenbins);

  //init_bounded_dev_vector log_len_cv_dev(1,nages,-2,2,3)
  vector len_sd(1,nages);
  vector len_cv(1,nages); //for fishgraph
  //All Fishery-dependent
  vector len_sd_L(1,nages);
  vector len_cv_L(1,nages); //for fishgraph
  //20 in size limit
  vector len_sd_20(1,nages);
  vector len_cv_20(1,nages);

//----Predicted length and age compositions
  matrix pred_cH_lenc(1,nyr_cH_lenc,1,nlenbins);
  matrix pred_cH_D_lenc(1,nyr_cH_D_lenc,1,nlenbins);
  matrix pred_HB_D_lenc(1,nyr_HB_D_lenc,1,nlenbins);  //***KIS***
```

```
  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_HB_agec(1,nyr_HB_agec,1,nages_agec_HB);
  matrix pred_HB_agec_allages(1,nyr_HB_agec,1,nages);
  matrix ErrorFree_HB_agec(1,nyr_HB_agec,1,nages);
  matrix pred_CVT_agec(1,nyr_CVT_agec,1,nages_agec);  //***KIS***
  matrix pred_CVT_agec_allages(1,nyr_CVT_agec,1,nages);  //***KIS***
  matrix ErrorFree_CVT_agec(1,nyr_CVT_agec,1,nages);  //***KIS***
  matrix pred_GR_agec(1,nyr_GR_agec,1,nages_agec);  //***KIS***
  matrix pred_GR_agec_allages(1,nyr_GR_agec,1,nages);  //***KIS***
  matrix ErrorFree_GR_agec(1,nyr_GR_agec,1,nages);  //***KIS***

//effective sample size applied in multinomial distributions
  vector nsamp_cH_lenc_allyr(styr,endyr);
  vector nsamp_cH_D_lenc_allyr(styr,endyr);  //***KIS***
  vector nsamp_HB_D_lenc_allyr(styr,endyr);  //***KIS***
  vector nsamp_cH_agec_allyr(styr,endyr);
  vector nsamp_HB_agec_allyr(styr,endyr);
  vector nsamp_CVT_agec_allyr(styr,endyr);  //***KIS***
  vector nsamp_GR_agec_allyr(styr,endyr);  //***KIS***

//Nfish used in MCB analysis (not used in fitting)
  vector nfish_cH_lenc_allyr(styr,endyr);
  vector nfish_cH_D_lenc_allyr(styr,endyr);   //***KIS***
  vector nfish_HB_D_lenc_allyr(styr,endyr);   //***KIS***
  vector nfish_cH_agec_allyr(styr,endyr);
  vector nfish_HB_agec_allyr(styr,endyr);
  vector nfish_CVT_agec_allyr(styr,endyr);   //***KIS***
```

```
  vector nfish_GR_agec_allyr(styr,endyr);   //***KIS***

//Computed effective sample size for output (not used in fitting)
  vector neff_cH_lenc_allyr_out(styr,endyr);
  vector neff_HB_D_lenc_allyr_out(styr,endyr);  //***KIS***
  vector neff_cH_agec_allyr_out(styr,endyr);
  vector neff_HB_agec_allyr_out(styr,endyr);
  vector neff_CVT_agec_allyr_out(styr,endyr);  //***KIS***
  vector neff_GR_agec_allyr_out(styr,endyr);  //***KIS***

//-----Population---------------------------------------------------------------------------
  matrix N(styr,endyr+1,1,nages);           //Population numbers by year and age at start of yr
  matrix N_mdyr(styr,endyr,1,nages);        //Population numbers by year and age at mdpt of yr: used for comps and
cpue
  matrix N_spawn(styr,endyr,1,nages);     //Population numbers by year and age at peaking spawning: used for SSB
  init_bounded_vector log_Nage_dev(2,nages,log_Nage_dev_LO,log_Nage_dev_HI,log_Nage_dev_PH);
  vector log_Nage_dev_output(1,nages);      //used in output. equals zero for first age
  matrix B(styr,endyr+1,1,nages);           //Population biomass by year and age at start of yr
  vector totB(styr,endyr+1);                //Total biomass by year
  vector totN(styr,endyr+1);                //Total abundance by year
  vector SSB(styr,endyr);                  //Total spawning biomass by year (female + male mature biomass)
  vector rec(styr,endyr+1);                 //Recruits by year
  vector prop_f(1,nages);
  vector maturity_f(1,nages);
  vector reprod(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  //KC comment out; this should fix it at the initial value

  vector rec_sigma_out(1,8);

  init_bounded_number R_autocorr(R_autocorr_LO,R_autocorr_HI,R_autocorr_PH);  //autocorrelation in SR  KC commented out since not estimated

  vector R_autocorr_out(1,8);


  number rec_sigma_sq;                    //square of rec_sigma

  number rec_logL_add;                    //additive term in -logL term


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

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

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


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

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


  number BiasCor;                         //Bias correction in equilibrium recruits

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

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

  number R1;                              //Recruits in styr

  number R_virgin;                        //unfished recruitment with bias correction

  vector SdS0(styr,endyr);                  //removed the +1
```

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

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

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

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

  init_bounded_number selpar_A50_cH1(selpar_A50_cH1_LO,selpar_A50_cH1_HI,selpar_A50_cH1_PH);

  init_bounded_number selpar_slope_cH1(selpar_slope_cH1_LO,selpar_slope_cH1_HI,selpar_slope_cH1_PH);

  init_bounded_number selpar_A50_cH2(selpar_A50_cH2_LO,selpar_A50_cH2_HI,selpar_A50_cH2_PH);

  init_bounded_number selpar_slope_cH2(selpar_slope_cH2_LO,selpar_slope_cH2_HI,selpar_slope_cH2_PH);

  init_bounded_number selpar_A50_cH3(selpar_A50_cH3_LO,selpar_A50_cH3_HI,selpar_A50_cH3_PH);

  init_bounded_number selpar_slope_cH3(selpar_slope_cH3_LO,selpar_slope_cH3_HI,selpar_slope_cH3_PH);


  vector selpar_A50_cH1_out(1,8);

  vector selpar_slope_cH1_out(1,8);

  vector selpar_A50_cH2_out(1,8);

  vector selpar_slope_cH2_out(1,8);

  vector selpar_A50_cH3_out(1,8);

  vector selpar_slope_cH3_out(1,8);


//Recreational ----------------------------------------------

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

  vector sel_HB_block1(1,nages);

  vector sel_HB_block2(1,nages);

  vector sel_HB_block3(1,nages);

  init_bounded_number selpar_A50_HB1(selpar_A50_HB1_LO,selpar_A50_HB1_HI,selpar_A50_HB1_PH);

  init_bounded_number selpar_slope_HB1(selpar_slope_HB1_LO,selpar_slope_HB1_HI,selpar_slope_HB1_PH);

  init_bounded_number selpar_A502_HB1(selpar_A502_HB1_LO,selpar_A502_HB1_HI,selpar_A502_HB1_PH);

  init_bounded_number
selpar_slope2_HB1(selpar_slope2_HB1_LO,selpar_slope2_HB1_HI,selpar_slope2_HB1_PH);
```

```
  init_bounded_number selpar_A50_HB2(selpar_A50_HB2_LO,selpar_A50_HB2_HI,selpar_A50_HB2_PH);
  init_bounded_number selpar_slope_HB2(selpar_slope_HB2_LO,selpar_slope_HB2_HI,selpar_slope_HB2_PH);
  init_bounded_number selpar_A502_HB2(selpar_A502_HB2_LO,selpar_A502_HB2_HI,selpar_A502_HB2_PH);
  init_bounded_number
selpar_slope2_HB2(selpar_slope2_HB2_LO,selpar_slope2_HB2_HI,selpar_slope2_HB2_PH);

  init_bounded_number selpar_A50_HB3(selpar_A50_HB3_LO,selpar_A50_HB3_HI,selpar_A50_HB3_PH);
  init_bounded_number selpar_slope_HB3(selpar_slope_HB3_LO,selpar_slope_HB3_HI,selpar_slope_HB3_PH);
  init_bounded_number selpar_A502_HB3(selpar_A502_HB3_LO,selpar_A502_HB3_HI,selpar_A502_HB3_PH);
  init_bounded_number
selpar_slope2_HB3(selpar_slope2_HB3_LO,selpar_slope2_HB3_HI,selpar_slope2_HB3_PH);

  vector selpar_A50_HB1_out(1,8);
  vector selpar_slope_HB1_out(1,8);
  vector selpar_A502_HB1_out(1,8);
  vector selpar_slope2_HB1_out(1,8);
  vector selpar_A50_HB2_out(1,8);
  vector selpar_slope_HB2_out(1,8);
  vector selpar_A502_HB2_out(1,8);
  vector selpar_slope2_HB2_out(1,8);
  vector selpar_A50_HB3_out(1,8);
  vector selpar_slope_HB3_out(1,8);
  vector selpar_A502_HB3_out(1,8);
  vector selpar_slope2_HB3_out(1,8);

  //General Rec in the last two time blocks  ***KIS
  matrix sel_GR(styr,endyr,1,nages);
  vector sel_GR_block1(1,nages);
  vector sel_GR_block2(1,nages);
  vector sel_GR_block3(1,nages);
```

```
  init_bounded_number selpar_A50_GR2(selpar_A50_GR2_LO,selpar_A50_GR2_HI,selpar_A50_GR2_PH);
  init_bounded_number selpar_slope_GR2(selpar_slope_GR2_LO,selpar_slope_GR2_HI,selpar_slope_GR2_PH);
  init_bounded_number selpar_A502_GR2(selpar_A502_GR2_LO,selpar_A502_GR2_HI,selpar_A502_GR2_PH);
  init_bounded_number
selpar_slope2_GR2(selpar_slope2_GR2_LO,selpar_slope2_GR2_HI,selpar_slope2_GR2_PH);

  init_bounded_number selpar_A50_GR3(selpar_A50_GR3_LO,selpar_A50_GR3_HI,selpar_A50_GR3_PH);
  init_bounded_number selpar_slope_GR3(selpar_slope_GR3_LO,selpar_slope_GR3_HI,selpar_slope_GR3_PH);
  //init_bounded_number selpar_A502_GR3(selpar_A502_GR3_LO,selpar_A502_GR3_HI,selpar_A502_GR3_PH);
  //init_bounded_number
selpar_slope2_GR3(selpar_slope2_GR3_LO,selpar_slope2_GR3_HI,selpar_slope2_GR3_PH);

  vector selpar_A50_GR2_out(1,8);
  vector selpar_slope_GR2_out(1,8);
  vector selpar_A502_GR2_out(1,8);
  vector selpar_slope2_GR2_out(1,8);

  vector selpar_A50_GR3_out(1,8);
  vector selpar_slope_GR3_out(1,8);
  //vector selpar_A502_GR3_out(1,8);
  //vector selpar_slope2_GR3_out(1,8);

//Discard selectivities
  matrix sel_cH_D(styr,endyr,1,nages);
  matrix sel_HB_D(styr,endyr,1,nages);  //***KIS GR uses HB discard selex
  vector sel_HB_D_block3(1,nages);
  matrix sel_GR_D(styr,endyr,1,nages);
```

```
 init_bounded_number
selpar_A50_HB2_D(selpar_A50_HB2_D_LO,selpar_A50_HB2_D_HI,selpar_A50_HB2_D_PH);

 init_bounded_number
selpar_slope_HB2_D(selpar_slope_HB2_D_LO,selpar_slope_HB2_D_HI,selpar_slope_HB2_D_PH);

 init_bounded_number
selpar_A502_HB2_D(selpar_A502_HB2_D_LO,selpar_A502_HB2_D_HI,selpar_A502_HB2_D_PH);

 init_bounded_number
selpar_slope2_HB2_D(selpar_slope2_HB2_D_LO,selpar_slope2_HB2_D_HI,selpar_slope2_HB2_D_PH);


 init_bounded_number
selpar_A50_HB3_D(selpar_A50_HB3_D_LO,selpar_A50_HB3_D_HI,selpar_A50_HB3_D_PH);

 init_bounded_number
selpar_slope_HB3_D(selpar_slope_HB3_D_LO,selpar_slope_HB3_D_HI,selpar_slope_HB3_D_PH);

 init_bounded_number
selpar_A502_HB3_D(selpar_A502_HB3_D_LO,selpar_A502_HB3_D_HI,selpar_A502_HB3_D_PH);

 init_bounded_number
selpar_slope2_HB3_D(selpar_slope2_HB3_D_LO,selpar_slope2_HB3_D_HI,selpar_slope2_HB3_D_PH);


 init_bounded_number
selpar_A50_cH2_D(selpar_A50_cH2_D_LO,selpar_A50_cH2_D_HI,selpar_A50_cH2_D_PH);

 init_bounded_number
selpar_slope_cH2_D(selpar_slope_cH2_D_LO,selpar_slope_cH2_D_HI,selpar_slope_cH2_D_PH);

 init_bounded_number
selpar_A502_cH2_D(selpar_A502_cH2_D_LO,selpar_A502_cH2_D_HI,selpar_A502_cH2_D_PH);

 init_bounded_number
selpar_slope2_cH2_D(selpar_slope2_cH2_D_LO,selpar_slope2_cH2_D_HI,selpar_slope2_cH2_D_PH);


 init_bounded_number
selpar_A50_cH3_D(selpar_A50_cH3_D_LO,selpar_A50_cH3_D_HI,selpar_A50_cH3_D_PH);

 init_bounded_number
selpar_slope_cH3_D(selpar_slope_cH3_D_LO,selpar_slope_cH3_D_HI,selpar_slope_cH3_D_PH);


 vector selpar_A50_HB2_D_out(1,8);

 vector selpar_slope_HB2_D_out(1,8);

 vector selpar_A502_HB2_D_out(1,8);

 vector selpar_slope2_HB2_D_out(1,8);
```

```
  vector selpar_A50_HB3_D_out(1,8);
  vector selpar_slope_HB3_D_out(1,8);
  vector selpar_A502_HB3_D_out(1,8);
  vector selpar_slope2_HB3_D_out(1,8);

  vector selpar_A50_cH2_D_out(1,8);
  vector selpar_slope_cH2_D_out(1,8);
  vector selpar_A502_cH2_D_out(1,8);
  vector selpar_slope2_cH2_D_out(1,8);

  vector selpar_A50_cH3_D_out(1,8);
  vector selpar_slope_cH3_D_out(1,8);

//Index selectivity (video uses same selex)  ***KIS***
  matrix sel_CVT(styr,endyr,1,nages);

  init_bounded_number selpar_A50_CVT(selpar_A50_CVT_LO,selpar_A50_CVT_HI,selpar_A50_CVT_PH);
  init_bounded_number selpar_slope_CVT(selpar_slope_CVT_LO,selpar_slope_CVT_HI,selpar_slope_CVT_PH);

  vector selpar_A50_CVT_out(1,8);
  vector selpar_slope_CVT_out(1,8);

//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_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_HB_D_cpue(styr_HB_D_cpue,endyr_HB_D_cpue);            //predicted HB disc index (number fish per
effort)

  matrix N_HB_D(styr_HB_D_cpue,endyr_HB_D_cpue,1,nages);                //used to compute HB disc index

  vector pred_CVT_cpue(styr_CVT_cpue,endyr_CVT_cpue);                //predicted SERFS index (number fish per
effort)

  matrix N_CVT(styr_CVT_cpue,endyr_CVT_cpue,1,nages);                //used to compute SERFS 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_HB(log_q_HB_LO,log_q_HB_HI,log_q_HB_PH);

  init_bounded_number log_q_HB_D(log_q_HB_D_LO,log_q_HB_D_HI,log_q_HB_D_PH);

  init_bounded_number log_q_CVT(log_q_CVT_LO,log_q_CVT_HI,log_q_CVT_PH);  //***KIS

  vector log_q_cH_out(1,8);

  vector log_q_HB_out(1,8);

  vector log_q_HB_D_out(1,8);

  vector log_q_CVT_out(1,8);  //***KIS


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

  vector q_rate_fcn_HB_D(styr_HB_D_cpue,endyr_HB_D_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 so commented out and declared as number (below)

  number q_DD_beta;

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

  number B0_q_DD;                 //B0 of ages q_DD_age plus

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


//Fishery dependent random walk catchability

 vector q_RW_log_dev_cH(styr_cH_cpue,endyr_cH_cpue-1);

 vector q_RW_log_dev_HB(styr_HB_cpue,endyr_HB_cpue-1);

 vector q_RW_log_dev_HB_D(styr_HB_D_cpue,endyr_HB_D_cpue-1);

 vector q_RW_log_dev_CVT(styr_CVT_cpue,endyr_CVT_cpue-1);


//Fishery dependent catchability over time, may be constant

 vector q_cH(styr_cH_cpue,endyr_cH_cpue);

 vector q_HB(styr_HB_cpue,endyr_HB_cpue);

 vector q_HB_D(styr_HB_D_cpue,endyr_HB_D_cpue);

 vector q_CVT(styr_CVT_cpue,endyr_CVT_cpue); //***KIS


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

//---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


  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 whole 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 whole 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_cH_num(styr,endyr,1,nages);            //discards (numbers) at age
  matrix D_cH_klb(styr,endyr,1,nages);            //discards (1000 lb whole) at age
  vector pred_cH_D_knum(styr_cH_D,endyr_cH_D);     //yearly dead discards summed over ages
  vector obs_cH_D(styr_cH_D,endyr_cH_D);          //observed releases multiplied by discard mortality
  vector pred_cH_D_klb(styr_cH_D,endyr_cH_D);      //yearly dead discards in klb whole wgt summed over ages

  matrix D_HB_num(styr,endyr,1,nages);            //discards (numbers) at age
  matrix D_HB_klb(styr,endyr,1,nages);            //discards (1000 lb) 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) 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 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_cH1;
 number Dmort_HB1;
 number Dmort_GR1;
 number Dmort_cH2;
 number Dmort_HB2;
 number Dmort_GR2;

////---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_cH_D_prop;    //proportion of F_sum attributable to cH discards, last X=selpar_n_yrs_wgted yrs
 number F_HB_D_prop;    //proportion of F_sum attributable to HB, last X=selpar_n_yrs_wgted yrs
 number F_GR_D_prop;    //proportion of F_sum attributable to GR, last X=selpar_n_yrs_wgted yrs

 number F_init_cH_prop;  //proportion of F_init attributable to cH, first X yrs, No diving or 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_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
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;
```

```
  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);
  vector SdSSB_msy(styr,endyr);
  number SdSSB_msy_end;
  number FdF_msy_end;
  number FdF_msy_end_mean;           //geometric mean of last X yrs
  vector SdSSB_F30(styr,endyr);
  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;                                //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------------------------------------------------------------------------

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

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

  vector M_constant_out(1,8);

  number smsy2msst;                       //scales 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_cH_D(log_avg_F_cH_D_LO,log_avg_F_cH_D_HI,log_avg_F_cH_D_PH);
  vector log_avg_F_cH_D_out(1,8);
  init_bounded_dev_vector
log_F_dev_cH_D(styr_cH_D,endyr_cH_D,log_F_dev_cH_D_LO,log_F_dev_cH_D_HI,log_F_dev_cH_D_PH);
  vector log_F_dev_cH_D_out(styr_cH_D,endyr_cH_D);
```

```
  matrix F_cH_D(styr,endyr,1,nages);
  vector F_cH_D_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_cH_D;

  init_bounded_number log_avg_F_HB_D(log_avg_F_HB_D_LO,log_avg_F_HB_D_HI,log_avg_F_HB_D_PH);
  vector log_avg_F_HB_D_out(1,8);
  init_bounded_dev_vector
log_F_dev_HB_D(styr_HB_D,endyr_HB_D,log_F_dev_HB_D_LO,log_F_dev_HB_D_HI,log_F_dev_HB_D_PH);
  vector log_F_dev_HB_D_out(styr_HB_D,endyr_HB_D);
  matrix F_HB_D(styr,endyr,1,nages);
  vector F_HB_D_out(styr,endyr);        //used for intermediate calculations in fcn get_mortality
  number log_F_dev_end_HB_D;

  init_bounded_number 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_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

//---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);    //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)-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_cH_D;
  number sdnr_lc_HB;
  number sdnr_lc_HB_D;  //***KIS***
  number sdnr_lc_CVT;  //***KIS***
  number sdnr_lc_GR;  //***KIS***
```

```
  number sdnr_ac_cH;
  number sdnr_ac_HB;
  number sdnr_ac_CVT;  //***KIS***
  number sdnr_ac_GR;  //***KIS***

  number sdnr_I_cH;
  number sdnr_I_HB;
  number sdnr_I_HB_D;
  number sdnr_I_CVT;  //***KIS

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

  number w_I_cH;
  number w_I_HB;
  number w_I_HB_D;
  number w_I_CVT; //**KIS

  number w_lc_cH;
  number w_lc_cH_D; //***KIS***
  //number w_lc_HB;
  number w_lc_HB_D;  //***KIS***
  //number w_lc_CVT;  //***KIS***
  //number w_lc_GR;  //***KIS***

  number w_ac_cH;
  number w_ac_HB;
```

```
number w_ac_CVT;  //***KIS***
number w_ac_GR;  //***KIS***

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_cH_D;
number f_HB_D;
number f_GR_D;

number f_cH_cpue;
number f_HB_cpue;
number f_HB_D_cpue;
number f_CVT_cpue;  //***KIS
number f_VID_cpue;  //***KIS

number f_cH_lenc;
number f_cH_D_lenc;  //***KIS***
number f_HB_lenc;
number f_HB_D_lenc;  //***KIS***
number f_CVT_lenc;  //***KIS***
```

```
  number f_GR_lenc;  //***KIS***

  number f_cH_agec;
  number f_HB_agec;
  number f_CVT_agec;  //***KIS***
  number f_GR_agec;  //***KIS

//  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; 10000; 10000;
 convergence_criteria 1e-2, 1e-2,1e-3, 1e-4; 1e-4; 1e-4;

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

// Set values of fixed parameters or set initial guess of estimated parameters
  Dmort_cH1=set_Dmort_cH1; Dmort_HB1=set_Dmort_HB1; Dmort_GR1=set_Dmort_GR1;
  Dmort_cH2=set_Dmort_cH2; Dmort_HB2=set_Dmort_HB2; Dmort_GR2=set_Dmort_GR2;

  for(iyear=styr_cH_D; iyear<=endyr_cH_D1; iyear++)
            {obs_cH_D(iyear)=Dmort_cH1*obs_cH_released(iyear);
            }

  for(iyear=endyr_cH_D1+1; iyear<=endyr_cH_D; iyear++)
            {obs_cH_D(iyear)=Dmort_cH2*obs_cH_released(iyear);
            }
```

```
for(iyear=styr_HB_D; iyear<=endyr_HB_D1; iyear++)
        {obs_HB_D(iyear)=Dmort_HB1*obs_HB_released(iyear);
        }
for(iyear=endyr_HB_D1+1; iyear<=endyr_HB_D; iyear++)
        {obs_HB_D(iyear)=Dmort_HB2*obs_HB_released(iyear);
        }

for(iyear=styr_GR_D; iyear<=endyr_GR_D1; iyear++)
        {obs_GR_D(iyear)=Dmort_GR1*obs_GR_released(iyear);
        }
for(iyear=endyr_GR_D1+1; iyear<=endyr_GR_D; iyear++)
        {obs_GR_D(iyear)=Dmort_GR2*obs_GR_released(iyear);
        }
//Population
Linf=set_Linf(1);
K=set_K(1);
t0=set_t0(1);
len_cv_val=set_len_cv(1);
//All fisheries
Linf_L=set_Linf_L(1);  //***KIS
K_L=set_K_L(1);
t0_L=set_t0_L(1);
len_cv_val_L=set_len_cv_L(1);
//20" size limit
Linf_20=set_Linf_20(1);  //***KIS
K_20=set_K_20(1);
t0_20=set_t0_20(1);
len_cv_val_20=set_len_cv_20(1);
```

```
M=set_M;
M_constant=set_M_constant(1);
smsy2msst=1.0-M_constant;
smsy2msst75=0.75;

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_HB=set_log_q_HB(1);
log_q_HB_D=set_log_q_HB_D(1);
log_q_CVT=set_log_q_CVT(1);

q_rate=set_q_rate;
q_rate_fcn_cH=1.0;
q_rate_fcn_HB=1.0;
q_rate_fcn_HB_D=1.0;
q_DD_beta=set_q_DD_beta;
q_DD_fcn=1.0;

q_RW_log_dev_cH.initialize();
q_RW_log_dev_HB.initialize();
q_RW_log_dev_HB_D.initialize();
q_RW_log_dev_CVT.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_HB_D_cpue; iyear<=endyr_HB_D_cpue; iyear++)
    {   if (iyear>styr_HB_D_cpue & iyear <=2003)
        {//q_rate_fcn_HB_D(iyear)=(1.0+q_rate)*q_rate_fcn_HB_D(iyear-1); //compound
          q_rate_fcn_HB_D(iyear)=(1.0+(iyear-styr_HB_D_cpue)*q_rate)*q_rate_fcn_HB_D(styr_HB_D_cpue);
//linear
        }
        if (iyear>2003) {q_rate_fcn_HB_D(iyear)=q_rate_fcn_HB_D(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_HB=set_w_I_HB;
 w_I_HB_D=set_w_I_HB_D;
 w_I_CVT=set_w_I_CVT;

 w_lc_cH=set_w_lc_cH;
 w_lc_cH_D=set_w_lc_cH_D;  //***KIS***
// w_lc_HB=set_w_lc_HB;
 w_lc_HB_D=set_w_lc_HB_D;  //***KIS***
// w_lc_CVT=set_w_lc_CVT;  //***KIS***
// w_lc_GR=set_w_lc_GR;  //***KIS***

 w_ac_cH=set_w_ac_cH;
 w_ac_HB=set_w_ac_HB;
 w_ac_CVT=set_w_ac_CVT;  //***KIS***
 w_ac_GR=set_w_ac_GR;  //***KIS***

 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_cH_D=set_log_avg_F_cH_D(1);
```

```
log_avg_F_HB_D=set_log_avg_F_HB_D(1);
log_avg_F_GR_D=set_log_avg_F_GR_D(1);

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_cH_D=set_log_F_dev_cH_D_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_A50_cH1=set_selpar_A50_cH1(1);
selpar_slope_cH1=set_selpar_slope_cH1(1);
selpar_A50_cH2=set_selpar_A50_cH2(1);
selpar_slope_cH2=set_selpar_slope_cH2(1);
selpar_A50_cH3=set_selpar_A50_cH3(1);
selpar_slope_cH3=set_selpar_slope_cH3(1);

selpar_A50_HB1=set_selpar_A50_HB1(1);
selpar_slope_HB1=set_selpar_slope_HB1(1);
selpar_A502_HB1=set_selpar_A502_HB1(1);
selpar_slope2_HB1=set_selpar_slope2_HB1(1);
selpar_A50_HB2=set_selpar_A50_HB2(1);
selpar_slope_HB2=set_selpar_slope_HB2(1);
selpar_A502_HB2=set_selpar_A502_HB2(1);
selpar_slope2_HB2=set_selpar_slope2_HB2(1);
selpar_A50_HB3=set_selpar_A50_HB3(1);
selpar_slope_HB3=set_selpar_slope_HB3(1);
selpar_A502_HB3=set_selpar_A502_HB3(1);
selpar_slope2_HB3=set_selpar_slope2_HB3(1);
```

```
selpar_A50_GR2=set_selpar_A50_GR2(1);
selpar_slope_GR2=set_selpar_slope_GR2(1);
selpar_A502_GR2=set_selpar_A502_GR2(1);
selpar_slope2_GR2=set_selpar_slope2_GR2(1);

selpar_A50_GR3=set_selpar_A50_GR3(1);
selpar_slope_GR3=set_selpar_slope_GR3(1);
//selpar_A502_GR3=set_selpar_A502_GR3(1);
//selpar_slope2_GR3=set_selpar_slope2_GR3(1);

selpar_A50_HB2_D=set_selpar_A50_HB2_D(1);
selpar_slope_HB2_D=set_selpar_slope_HB2_D(1);
selpar_A502_HB2_D=set_selpar_A502_HB2_D(1);
selpar_slope2_HB2_D=set_selpar_slope2_HB2_D(1);

selpar_A50_HB3_D=set_selpar_A50_HB3_D(1);
selpar_slope_HB3_D=set_selpar_slope_HB3_D(1);
selpar_A502_HB3_D=set_selpar_A502_HB3_D(1);
selpar_slope2_HB3_D=set_selpar_slope2_HB3_D(1);

selpar_A50_cH2_D=set_selpar_A50_cH2_D(1);
selpar_slope_cH2_D=set_selpar_slope_cH2_D(1);
selpar_A502_cH2_D=set_selpar_A502_cH2_D(1);
selpar_slope2_cH2_D=set_selpar_slope2_cH2_D(1);

selpar_A50_cH3_D=set_selpar_A50_cH3_D(1);
selpar_slope_cH3_D=set_selpar_slope_cH3_D(1);
```

```
 selpar_A50_CVT=set_selpar_A50_CVT(1);
 selpar_slope_CVT=set_selpar_slope_CVT(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;

fecpar_a=set_fecpar_a;
fecpar_b=set_fecpar_b;
fecpar_batches=set_fecpar_batches;

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

```
nsamp_cH_lenc_allyr=missing;
        nsamp_cH_D_lenc_allyr=missing;  //***KIS***
nsamp_HB_D_lenc_allyr=missing;  //***KIS***
nsamp_cH_agec_allyr=missing;
nsamp_HB_agec_allyr=missing;
        nsamp_CVT_agec_allyr=missing;  //***KIS
        nsamp_GR_agec_allyr=missing;  //***KIS

nfish_cH_lenc_allyr=missing;
        nfish_cH_D_lenc_allyr=missing;  //***KIS***
nfish_HB_D_lenc_allyr=missing;  //***KIS***
nfish_cH_agec_allyr=missing;
nfish_HB_agec_allyr=missing;
        nfish_CVT_agec_allyr=missing;  //***KIS***
        nfish_GR_agec_allyr=missing;  //***KIS***

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_cH_D_lenc; iyear++)                    //***KIS***
  {if (nsamp_cH_D_lenc(iyear)>=minSS_cH_D_lenc)
    {nsamp_cH_D_lenc_allyr(yrs_cH_D_lenc(iyear))=nsamp_cH_D_lenc(iyear);
     nfish_cH_D_lenc_allyr(yrs_cH_D_lenc(iyear))=nfish_cH_D_lenc(iyear);}}
for (iyear=1; iyear<=nyr_HB_D_lenc; iyear++)                    //***KIS***
  {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_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_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_CVT_agec; iyear++)  //***KIS***
        {if (nsamp_CVT_agec(iyear)>=minSS_CVT_agec)
         {nsamp_CVT_agec_allyr(yrs_CVT_agec(iyear))=nsamp_CVT_agec(iyear);
          nfish_CVT_agec_allyr(yrs_CVT_agec(iyear))=nfish_CVT_agec(iyear);}}
      for (iyear=1; iyear<=nyr_GR_agec; iyear++)  //***KIS***
        {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_cH_D.initialize(); D_cH_num.initialize();
```

```
F_HB_D.initialize(); D_HB_num.initialize();
F_GR_D.initialize(); D_GR_num.initialize();

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



sel_cH.initialize();
sel_HB.initialize();
sel_GR.initialize();
sel_cH_D.initialize();
sel_HB_D.initialize();
sel_GR_D.initialize();
sel_CVT.initialize();

sel_HB_block1.initialize();
sel_HB_block2.initialize();
sel_HB_block3.initialize();
sel_GR_block1.initialize();
sel_GR_block2.initialize();
sel_GR_block3.initialize();
sel_HB_D_block3.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;
  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

        //population total length in mm

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

   meanlen_TL=Linf*(1.0-mfexp(-K*(agebins-t0+0.5)));

   wgt_kg=wgtpar_a*pow(meanlen_TL,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*pow(meanlen_TL,fecpar_b)/fecpar_scale;  //annual egg production of a mature female at age
in units of fecpar_scale


        //20 in size limit

        meanlen_TL_20=Linf_20*(1.0-mfexp(-K_20*(agebins-t0_20+0.5)));    //Landings total length in mm

   wgt_kg_20=wgtpar_a*pow(meanlen_TL_20,wgtpar_b);          //whole wgt in kg

   wgt_g_20=wgt_kg_20/g2kg;                                 //convert wgt in kg to weight in g

   wgt_mt_20=wgt_g_20*g2mt;                                 //convert weight in g to weight in mt

   wgt_klb_20=mt2klb*wgt_mt_20;                             //1000 lb of whole wgt

   wgt_lb_20=mt2lb*wgt_mt_20;                               //1000 lb of whole wgt


        //All fisheries

        meanlen_TL_L=Linf_L*(1.0-mfexp(-K_L*(agebins-t0_L+0.5)));    //Landings total length in mm

   wgt_kg_L=wgtpar_a*pow(meanlen_TL_L,wgtpar_b);          //whole wgt in kg

   wgt_g_L=wgt_kg_L/g2kg;                                 //convert wgt in kg to weight in g

   wgt_mt_L=wgt_g_L*g2mt;                                 //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


FUNCTION get_reprod
```

```
        reprod=elem_prod(elem_prod(elem_prod(prop_f,maturity_f),fecundity),fecpar_batches);

FUNCTION get_length_at_age_dist
  //compute matrix of length at age, based on the normal distribution
   //population
        for (iage=1;iage<=nages;iage++)
  {len_cv(iage)=len_cv_val;
   len_sd(iage)=meanlen_TL(iage)*len_cv(iage);
        zscore_lzero=(0.0-meanlen_TL(iage))/len_sd(iage);
        cprob_lzero=cumd_norm(zscore_lzero);

        //20 in size limit
        //len_cv_20(iage)=mfexp(log_len_cv_20+log_len_cv_dev_20(iage));
   len_cv_20(iage)=len_cv_val_20;
   len_sd_20(iage)=meanlen_TL_20(iage)*len_cv_20(iage);
        zscore_lzero_20=(0.0-meanlen_TL_20(iage))/len_sd_20(iage);
        cprob_lzero_20=cumd_norm(zscore_lzero_20);

        //All fishery dependent
        //len_cv_L(iage)=mfexp(log_len_cv_L+log_len_cv_dev_L(iage));
   len_cv_L(iage)=len_cv_val_L;
   len_sd_L(iage)=meanlen_TL_L(iage)*len_cv_L(iage);
        zscore_lzero_L=(0.0-meanlen_TL_L(iage))/len_sd_L(iage);
        cprob_lzero_L=cumd_norm(zscore_lzero_L);

   //first length bin
        //population
    zscore_len=((lenbins(1)+0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
```

```
cprob_lenvec(1)=cumd_norm(zscore_len);          //includes any probability mass below zero
lenprob(iage,1)=cprob_lenvec(1)-cprob_lzero;    //removes any probability mass below zero

        //20 in size limit
         zscore_len_20=((lenbins(1)+0.5*lenbins_width)-meanlen_TL_20(iage)) / len_sd_20(iage);
cprob_lenvec_20(1)=cumd_norm(zscore_len_20);          //includes any probability mass below zero
lenprob_20(iage,1)=cprob_lenvec_20(1)-cprob_lzero_20;    //removes any probability mass below zero

        //All fishery dependent
zscore_len_L=((lenbins(1)+0.5*lenbins_width)-meanlen_TL_L(iage)) / len_sd_L(iage);
cprob_lenvec_L(1)=cumd_norm(zscore_len_L);          //includes any probability mass below zero
lenprob_L(iage,1)=cprob_lenvec_L(1)-cprob_lzero_L;    //removes any probability mass below zero

//most other length bins
//population
for (ilen=2;ilen<nlenbins;ilen++)
 {
  zscore_len=((lenbins(ilen)+0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
                cprob_lenvec(ilen)=cumd_norm(zscore_len);
  lenprob(iage,ilen)=cprob_lenvec(ilen)-cprob_lenvec(ilen-1);
 }

        //20 in size limit
        for (ilen=2;ilen<nlenbins;ilen++)
 {
  zscore_len_20=((lenbins(ilen)+0.5*lenbins_width)-meanlen_TL_20(iage)) / len_sd_20(iage);
                cprob_lenvec_20(ilen)=cumd_norm(zscore_len_20);
  lenprob_20(iage,ilen)=cprob_lenvec_20(ilen)-cprob_lenvec_20(ilen-1);
 }
```

```
            //All fishery dependent
    for (ilen=2;ilen<nlenbins;ilen++)
      {
        zscore_len_L=((lenbins(ilen)+0.5*lenbins_width)-meanlen_TL_L(iage)) / len_sd_L(iage);
                        cprob_lenvec_L(ilen)=cumd_norm(zscore_len_L);
        lenprob_L(iage,ilen)=cprob_lenvec_L(ilen)-cprob_lenvec_L(ilen-1);
      }
    //last length bin is a plus group
            //population
    zscore_len=((lenbins(nlenbins)-0.5*lenbins_width)-meanlen_TL(iage)) / len_sd(iage);
            lenprob(iage,nlenbins)=1.0-cumd_norm(zscore_len);
      lenprob(iage)=lenprob(iage)/(1.0-cprob_lzero);  //renormalize to account for any prob mass below size=0

            //20 in size limit
            zscore_len_20=((lenbins(nlenbins)-0.5*lenbins_width)-meanlen_TL_20(iage)) / len_sd_20(iage);
            lenprob_20(iage,nlenbins)=1.0-cumd_norm(zscore_len_20);
      lenprob_20(iage)=lenprob_20(iage)/(1.0-cprob_lzero_20);  //renormalize to account for any prob mass below
size=0

            //All fishery dependent
    zscore_len_L=((lenbins(nlenbins)-0.5*lenbins_width)-meanlen_TL_L(iage)) / len_sd_L(iage);
            lenprob_L(iage,nlenbins)=1.0-cumd_norm(zscore_len_L);
      lenprob_L(iage)=lenprob_L(iage)/(1.0-cprob_lzero_L);  //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_L;
  lenprob_cH_D=lenprob; //***KIS population
  lenprob_HB=lenprob_L;
```

```
 lenprob_HB_D=lenprob; //population
 lenprob_CVT=lenprob; //***KIS population
 lenprob_GR=lenprob_L; //***KIS

FUNCTION get_weight_at_age_landings  ///***in whole weight

 //for (iyear=styr; iyear<=endyr; iyear++)
 for (iyear=styr; iyear<=endyr_selex_phase2; iyear++) //start through 1991
 {
  len_cH_mm(iyear)=meanlen_TL_L;  //using all fishery dependent growth for fisheries until 20 in size limit
  wholewgt_cH_klb(iyear)=wgt_klb_L;
  len_HB_mm(iyear)=meanlen_TL_L;
  wholewgt_HB_klb(iyear)=wgt_klb_L;
  len_GR_mm(iyear)=meanlen_TL_L;
  wholewgt_GR_klb(iyear)=wgt_klb_L;

  len_cH_D_mm(iyear)=meanlen_TL;  //using the population growth to define discards
  wholewgt_cH_D_klb(iyear)=wgt_klb;
  len_HB_D_mm(iyear)=meanlen_TL;
  wholewgt_HB_D_klb(iyear)=wgt_klb;
  len_GR_D_mm(iyear)=meanlen_TL;
  wholewgt_GR_D_klb(iyear)=wgt_klb;
 }
 for (iyear=endyr_selex_phase2+1; iyear<=endyr_selex_phase3; iyear++)  //1992 through 2009
 {
  len_cH_mm(iyear)=meanlen_TL_20;
  wholewgt_cH_klb(iyear)=wgt_klb_20;
  len_HB_mm(iyear)=meanlen_TL_20;
  wholewgt_HB_klb(iyear)=wgt_klb_20;
```

```
    len_GR_mm(iyear)=meanlen_TL_20;
    wholewgt_GR_klb(iyear)=wgt_klb_20;

    len_cH_D_mm(iyear)=meanlen_TL;
    wholewgt_cH_D_klb(iyear)=wgt_klb;   //using population growth curve for the discard weight
    len_HB_D_mm(iyear)=meanlen_TL;
    wholewgt_HB_D_klb(iyear)=wgt_klb;
    len_GR_D_mm(iyear)=meanlen_TL;
    wholewgt_GR_D_klb(iyear)=wgt_klb;
  }
  for (iyear=endyr_selex_phase3+1; iyear<=endyr; iyear++)  //2010 through 2014 save in case we end up treating as
a different growth curve
  {
    len_cH_mm(iyear)=meanlen_TL_L;
    wholewgt_cH_klb(iyear)=wgt_klb_L;
    len_HB_mm(iyear)=meanlen_TL_L;
    wholewgt_HB_klb(iyear)=wgt_klb_L;
    len_GR_mm(iyear)=meanlen_TL_L;
    wholewgt_GR_klb(iyear)=wgt_klb_L;

    len_cH_D_mm(iyear)=meanlen_TL;
    wholewgt_cH_D_klb(iyear)=wgt_klb;
    len_HB_D_mm(iyear)=meanlen_TL;
    wholewgt_HB_D_klb(iyear)=wgt_klb;
    len_GR_D_mm(iyear)=meanlen_TL;
    wholewgt_GR_D_klb(iyear)=wgt_klb;
  }

FUNCTION get_spr_F0
```

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

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

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

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

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

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

  }

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

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


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

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


FUNCTION get_selectivity


  sel_HB_block1=logistic_double(agebins, selpar_A50_HB1, selpar_slope_HB1, selpar_A502_HB1,
selpar_slope2_HB1);

  sel_HB_block2=logistic_double(agebins, selpar_A50_HB2, selpar_slope_HB2,
selpar_A502_HB2,selpar_slope2_HB2);

  sel_HB_block3=logistic_double(agebins, selpar_A50_HB3, selpar_slope_HB3, selpar_A502_HB3,
selpar_slope2_HB3);

  sel_GR_block1=sel_HB_block1;

  sel_GR_block2=logistic_double(agebins, selpar_A50_GR2, selpar_slope_GR2,
selpar_A502_GR2,selpar_slope2_GR2);

  sel_HB_D_block3=logistic_double(agebins, selpar_A50_HB3_D, selpar_slope_HB3_D, selpar_A502_HB3_D,
selpar_slope2_HB3_D);

  sel_GR_block3=logistic(agebins, selpar_A50_GR3, selpar_slope_GR3);


  //sel_GR_block3=logistic_double(agebins, selpar_A50_GR3, selpar_slope_GR3, selpar_A502_GR3,
selpar_slope2_GR3);
```

```
 //BLOCK 1 for selex. No limit through to 20" size limit

 for (iyear=styr; iyear<=endyr_selex_phase2; iyear++)

  {

   sel_cH(iyear)=logistic(agebins, selpar_A50_cH1, selpar_slope_cH1);

   sel_HB(iyear)(1,nages_agec_HB)=sel_HB_block1(1,nages_agec_HB);

          sel_HB(iyear)((nages_agec_HB+1),nages)=sel_HB_block1(nages_agec_HB);

   sel_GR(iyear)(1,nages_agec_HB)=sel_GR_block1(1,nages_agec_HB); //NOTE: uses nages_agec_HB bc this
selex mirrors HB

          sel_GR(iyear)((nages_agec_HB+1),nages)=sel_GR_block1(nages_agec_HB);

          sel_cH_D(iyear)=logistic_double(agebins, selpar_A50_cH2_D, selpar_slope_cH2_D,
selpar_A502_cH2_D, selpar_slope2_cH2_D);

          sel_HB_D(iyear)=logistic_double(agebins, selpar_A50_HB2_D, selpar_slope_HB2_D,
selpar_A502_HB2_D, selpar_slope2_HB2_D);

  }


 //BLOCK 2 for selex. 20" size limit until 2010


 for (iyear=(endyr_selex_phase2+1); iyear<=endyr_selex_phase3; iyear++)

  {

   sel_cH(iyear)=logistic(agebins, selpar_A50_cH2, selpar_slope_cH2);

   sel_HB(iyear)(1,nages_agec_HB)=sel_HB_block2(1,nages_agec_HB);

          sel_HB(iyear)((nages_agec_HB+1),nages)=sel_HB_block2(nages_agec_HB);

   sel_GR(iyear)(1,nages_agec)=sel_GR_block2(1,nages_agec); //NOTE: uses nages_agec_HB bc this selex mirrors
HB

          sel_GR(iyear)((nages_agec+1),nages)=sel_GR_block2(nages_agec);

   sel_cH_D(iyear)=logistic_double(agebins, selpar_A50_cH2_D, selpar_slope_cH2_D, selpar_A502_cH2_D,
selpar_slope2_cH2_D);  //, selpar_init_cH2_D, selpar_final_cH2_D);

          sel_HB_D(iyear)=logistic_double(agebins, selpar_A50_HB2_D, selpar_slope_HB2_D,
selpar_A502_HB2_D, selpar_slope2_HB2_D);  //, selpar_init_HB2_D, selpar_final_HB2_D);

  }


 //BLOCK 3 for selex.  Moratorium and mini-seasons with no size limit during the mini-season
```

```
  for (iyear=(endyr_selex_phase3+1); iyear<=endyr; iyear++)
  {
   sel_cH(iyear)=logistic(agebins, selpar_A50_cH3, selpar_slope_cH3);
          sel_HB(iyear)(1,nages_agec_HB)=sel_HB_block3(1,nages_agec_HB);
          sel_HB(iyear)((nages_agec_HB+1),nages)=sel_HB_block3(nages_agec_HB);
   sel_GR(iyear)(1,nages_agec)=sel_GR_block3(1,nages_agec);
          sel_GR(iyear)((nages_agec+1),nages)=sel_GR_block3(nages_agec);
   sel_cH_D(iyear)=logistic(agebins, selpar_A50_cH3_D, selpar_slope_cH3_D);//, selpar_ascwid_cH3_D,
selpar_deswid_cH3_D, selpar_init_cH3_D, selpar_final_cH3_D);
          sel_HB_D(iyear)(1,nages_agec_HB)=sel_HB_D_block3(1,nages_agec_HB);
          sel_HB_D(iyear)((nages_agec_HB+1),nages)=sel_HB_D_block3(nages_agec_HB);
  }
   sel_GR_D=sel_HB_D;
  for (iyear=styr; iyear<=endyr; iyear++)
    {
     sel_CVT(iyear)=logistic(agebins, selpar_A50_CVT, selpar_slope_CVT);
    }

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)); //}
   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)); //}
   F_HB(iyear)=sel_HB(iyear)*F_HB_out(iyear);
   Fsum(iyear)+=F_HB_out(iyear);
}

if(iyear>=styr_GR_L & iyear<=endyr_GR_L)
{  F_GR_out(iyear)=mfexp(log_avg_F_GR+log_F_dev_GR(iyear)); //}
   F_GR(iyear)=sel_GR(iyear)*F_GR_out(iyear); //(general rec shares headboat selex) fixed ***KIS
   Fsum(iyear)+=F_GR_out(iyear);
}

if(iyear>=styr_cH_D & iyear<=endyr_cH_D)
{  F_cH_D_out(iyear)=mfexp(log_avg_F_cH_D+log_F_dev_cH_D(iyear)); //}
   F_cH_D(iyear)=sel_cH_D(iyear)*F_cH_D_out(iyear);
   Fsum(iyear)+=F_cH_D_out(iyear);
}

if(iyear>=styr_HB_D & iyear<=endyr_HB_D)
{  F_HB_D_out(iyear)=mfexp(log_avg_F_HB_D+log_F_dev_HB_D(iyear)); //}
   F_HB_D(iyear)=sel_HB_D(iyear)*F_HB_D_out(iyear);
   Fsum(iyear)+=F_HB_D_out(iyear);
}
```

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

  //Total F at age
  F(iyear)=F_cH(iyear);  //first in additive series (NO +=)
  F(iyear)+=F_HB(iyear);
  F(iyear)+=F_GR(iyear);
  F(iyear)+=F_cH_D(iyear);
  F(iyear)+=F_HB_D(iyear);
  F(iyear)+=F_GR_D(iyear);

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

 }  //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=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(lo
g_avg_F_GR+log_F_dev_init_GR);

  F_init_cH_prop= 1.0; //mfexp(log_avg_F_cH+log_F_dev_init_cH)/F_init_denom;  //Leave this machinery in, in
case a sensitivity is run with a later start year.

  F_init_HB_prop= 0.0; //mfexp(log_avg_F_HB+log_F_dev_init_HB)/F_init_denom;

  F_init_GR_prop= 0.0; //mfexp(log_avg_F_GR+log_F_dev_init_GR)/F_init_denom;


  F_initial=sel_cH(styr)*F_init*F_init_cH_prop+

          sel_HB(styr)*F_init*F_init_HB_prop+

          sel_GR(styr)*F_init*F_init_GR_prop; //(GR uses HB selex)  fixed ***KIS

  Z_initial=M+F_initial;


//Initial equilibrium age structure

  N_spr_initial(1)=1.0*mfexp(-1.0*Z_initial(1)*spawn_time_frac); //at peak spawning time;

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

    {

      N_spr_initial(iage)=N_spr_initial(iage-1)*

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

    }

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

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

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

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

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


//Compute equilibrium age structure for first year

  N_initial_eq(1)=R1;

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

  {

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

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

  }

  //plus group calculation

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


//Add deviations to initial equilibrium N

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


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

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


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

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


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

  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));


        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));

    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_cH_D; iyear<=endyr_cH_D; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      D_cH_num(iyear,iage)=N(iyear,iage)*F_cH_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
   pred_cH_D_knum(iyear)=sum(D_cH_num(iyear))/1000.0;            //pred annual dead discards in 1000s (for
matching data)
   pred_cH_D_klb(iyear)=sum(elem_prod(D_cH_num(iyear),wholewgt_cH_D_klb(iyear))); //annual dead discards
in 1000 lb whole (for output only)
  }

  for (iyear=styr_HB_D; iyear<=endyr_HB_D; iyear++)
  {
    for (iage=1; iage<=nages; iage++)
    {
      D_HB_num(iyear,iage)=N(iyear,iage)*F_HB_D(iyear,iage)*
        (1.-mfexp(-1.*Z(iyear,iage)))/Z(iyear,iage);
    }
   pred_HB_D_knum(iyear)=sum(D_HB_num(iyear))/1000.0;            //pred annual dead discards in 1000s (for
matHBing 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
matGRing 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_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_HB_D_cpue; iyear<=endyr_HB_D_cpue; iyear++)
    {   if (iyear>styr_HB_D_cpue & iyear <=2003)
        {//q_rate_fcn_HB_D(iyear)=(1.0+q_rate)*q_rate_fcn_HB_D(iyear-1); //compound
          q_rate_fcn_HB_D(iyear)=(1.0+(iyear-styr_HB_D_cpue)*q_rate)*q_rate_fcn_HB_D(styr_HB_D_cpue);
//linear
        }
        if (iyear>2003) {q_rate_fcn_HB_D(iyear)=q_rate_fcn_HB_D(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));
     pred_cH_cpue(iyear)=q_cH(iyear)*q_rate_fcn_cH(iyear)*q_DD_fcn(iyear)*sum(N_cH(iyear));
     if (iyear<endyr_cH_cpue){q_cH(iyear+1)=q_cH(iyear)*mfexp(q_RW_log_dev_cH(iyear));}
  }

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

 //HB disc cpue         ***KIS
  q_HB_D(styr_HB_D_cpue)=mfexp(log_q_HB_D);
  for (iyear=styr_HB_D_cpue; iyear<=endyr_HB_D_cpue; iyear++)
  {
     N_HB_D(iyear)=elem_prod(N_mdyr(iyear),sel_HB_D(endyr_selex_phase3));
     pred_HB_D_cpue(iyear)=q_HB_D(iyear)*q_rate_fcn_HB_D(iyear)*q_DD_fcn(iyear)*sum(N_HB_D(iyear));
     if (iyear<endyr_HB_D_cpue){q_HB_D(iyear+1)=q_HB_D(iyear)*mfexp(q_RW_log_dev_HB_D(iyear));}
  }
```

```
 //MARMAP CVT cpue
  q_CVT(styr_CVT_cpue)=mfexp(log_q_CVT);
  for (iyear=styr_CVT_cpue; iyear<=endyr_CVT_cpue; iyear++)
  {
   N_CVT(iyear)=elem_prod(N_mdyr(iyear),sel_CVT(iyear));
            pred_CVT_cpue(iyear)=q_CVT(iyear)*q_DD_fcn(iyear)*sum(N_CVT(iyear));
   if (iyear<endyr_CVT_cpue){q_CVT(iyear+1)=q_CVT(iyear)*mfexp(q_RW_log_dev_CVT(iyear));}
  }

FUNCTION get_length_comps

  //comm handline

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

  //comm discards   ***KIS
  for (iyear=1;iyear<=nyr_cH_D_lenc;iyear++)

{pred_cH_D_lenc(iyear)=(D_cH_num(yrs_cH_D_lenc(iyear))*lenprob_cH_D)/sum(D_cH_num(yrs_cH_D_lenc(iyear)));}

  //headboat discards   ***KIS
  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_lenc(iyear)));}

FUNCTION get_age_comps
```

```
 //Commercial handline
 for (iyear=1;iyear<=nyr_cH_agec;iyear++)
 {
  ErrorFree_cH_agec(iyear)=L_cH_num(yrs_cH_agec(iyear))/sum(L_cH_num(yrs_cH_agec(iyear)));
  //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
 }

 //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_HB; iage++) {pred_HB_agec(iyear,iage)=pred_HB_agec_allages(iyear,iage);}
  for (iage=(nages_agec_HB+1); iage<=nages; iage++)
{pred_HB_agec(iyear,nages_agec_HB)+=pred_HB_agec_allages(iyear,iage);} //plus group
 }

  //MARMAP CVT ***KIS
 for (iyear=1;iyear<=nyr_CVT_agec;iyear++)
 {
  ErrorFree_CVT_agec(iyear)=N_CVT(yrs_CVT_agec(iyear))/sum(N_CVT(yrs_CVT_agec(iyear)));
  pred_CVT_agec_allages(iyear)=age_error*ErrorFree_CVT_agec(iyear);
  for (iage=1; iage<=nages_agec; iage++) {pred_CVT_agec(iyear,iage)=pred_CVT_agec_allages(iyear,iage);}
  for (iage=(nages_agec+1); iage<=nages; iage++)
{pred_CVT_agec(iyear,nages_agec)+=pred_CVT_agec_allages(iyear,iage);} //plus group
 }
```

```
  //PR/CH Recreational  ***KIS
 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_cH_D+
       sum(log_F_dev_cH_D((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_cH_D_prop=mfexp((selpar_n_yrs_wgted*log_avg_F_cH_D+
      sum(log_F_dev_cH_D((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_cH_D=sum(log_F_dev_cH_D((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);   // (GR uses HB selex ) fixed ***KIS

 F_end_D=sel_cH_D(endyr)*mfexp(log_avg_F_cH_D+log_F_dev_end_cH_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);  // (GR uses HB selex )

 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;  //**KIS changed to whole weight
 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_cH_D_prop+F_HB_D_prop+F_GR_D_prop;  //**KIS changed to whole weight
 wgt_wgted_D_klb=F_cH_D_prop/wgt_wgted_D_denom*wholewgt_cH_D_klb(endyr)+
            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  ***KIS

 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));
  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 whole wgt
  }
}
spr_ratio=spr_spr/spr_F0;
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);
            }
  }
//--------------------------------------------------------------------------------------------------------------------------------------------------------
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_TL);
  len_cv_L=elem_div(len_sd_L,meanlen_TL_L);
  len_cv_20=elem_div(len_sd_20,meanlen_TL_20);

  //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();
 D_cH_klb.initialize();
 D_HB_klb.initialize();
 D_GR_klb.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_cH_D && iyear<=endyr_cH_D)
     {
      D_total_knum_yr(iyear)+=pred_cH_D_knum(iyear);
      D_total_klb_yr(iyear)+=pred_cH_D_klb(iyear);
      D_cH_klb(iyear)=elem_prod(D_cH_num(iyear),wholewgt_cH_D_klb(iyear));     //in 1000 lb
     }

     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));     //in 1000 lb
     }
```

```
    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));     //in 1000 lb
    }
}

L_total_num=L_cH_num+L_HB_num+L_GR_num;   //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_cH_num+D_HB_num+D_GR_num);          //discards at age in number fish
D_total_klb=D_cH_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));
rec=column(N,1);
SdS0=SSB/S0;

Fend_mean_temp=1.0;
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));
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;
   }
 if(SSB_msy_out>0)
   {
    SdSSB_msy=SSB/SSB_msy_out;
    SdSSB_msy_end=SdSSB_msy(endyr);
   }

          if(F30_out>0)
   {
            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/(smsy2msst75*SSB_F30_out);
    SdSSB_F30_end=SdSSB_F30(endyr);
            Sdmsst_F30_end=Sdmsst_F30(endyr);
   }
  //fill in log recruitment deviations for yrs they are nonzero
  for(iyear=styr_rec_dev; iyear<=endyr_rec_dev; iyear++)
    {log_rec_dev_output(iyear)=log_rec_dev(iyear);}
  //fill in log Nage deviations for ages they are nonzero (ages2+)
  for(iage=2; iage<=nages; iage++)
    {log_Nage_dev_output(iage)=log_Nage_dev(iage);}

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

```
FUNCTION get_effective_sample_sizes
    neff_cH_lenc_allyr_out=missing;
        neff_HB_D_lenc_allyr_out=missing;

    neff_cH_agec_allyr_out=missing;
    neff_HB_agec_allyr_out=missing;
        neff_CVT_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_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(iyear));}
       else {neff_HB_D_lenc_allyr_out(yrs_HB_D_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_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_CVT_agec; iyear++)  //***KIS
       {if (nsamp_CVT_agec(iyear)>=minSS_CVT_agec)
{neff_CVT_agec_allyr_out(yrs_CVT_agec(iyear))=multinom_eff_N(pred_CVT_agec(iyear),obs_CVT_agec(iyear))
;}
        else {neff_CVT_agec_allyr_out(yrs_CVT_agec(iyear))=-99;}
        }

           for (iyear=1; iyear<=nyr_GR_agec; iyear++)  //***KIS
       {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_HB_cpue=0.0;
  f_HB_cpue=lk_lognormal(pred_HB_cpue, obs_HB_cpue, HB_cpue_cv, w_I_HB);
  fval+=f_HB_cpue;
  fval_data+=f_HB_cpue;

  f_HB_D_cpue=0.0;
  f_HB_D_cpue=lk_lognormal(pred_HB_D_cpue, obs_HB_D_cpue, HB_D_cpue_cv, w_I_HB_D);
  fval+=f_HB_D_cpue;
  fval_data+=f_HB_D_cpue;

  f_CVT_cpue=0.0;
  f_CVT_cpue=lk_lognormal(pred_CVT_cpue, obs_CVT_cpue, CVT_cpue_cv, w_I_CVT);
  fval+=f_CVT_cpue;
  fval_data+=f_CVT_cpue;

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

  //f_cH_L in 1000 lb whole wgt  ***KIS changed to whole
  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_cH_D in 1000 fish
  f_cH_D=lk_lognormal(pred_cH_D_knum(styr_cH_D,endyr_cH_D), obs_cH_D(styr_cH_D,endyr_cH_D),
                cH_D_cv(styr_cH_D,endyr_cH_D), w_D);
  fval+=f_cH_D;
  fval_data+=f_cH_D;

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

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

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

  //f_cH_lenc
  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);
  fval+=f_cH_lenc;
  fval_data+=f_cH_lenc;

  //f_cH_D_lenc
  f_cH_D_lenc=lk_robust_multinomial(nsamp_cH_D_lenc, pred_cH_D_lenc, obs_cH_D_lenc, nyr_cH_D_lenc,
double(nlenbins), minSS_cH_D_lenc, w_lc_cH_D);
  fval+=f_cH_D_lenc;
  fval_data+=f_cH_D_lenc;//***KIS***

  //f_HB_D_lenc
  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);
  fval+=f_HB_D_lenc;
  fval_data+=f_HB_D_lenc;//***KIS***

//---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);
```

```
  fval+=f_cH_agec;

  fval_data+=f_cH_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_HB), minSS_HB_agec, w_ac_HB);

  fval+=f_HB_agec;

  fval_data+=f_HB_agec;


  //f_CVT_agec

  f_CVT_agec=lk_robust_multinomial(nsamp_CVT_agec, pred_CVT_agec, obs_CVT_agec, nyr_CVT_agec,
double(nages_agec), minSS_CVT_agec, w_ac_CVT);

  fval+=f_CVT_agec;

  fval_data+=f_CVT_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);

  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(len_cv_val,set_len_cv(5),set_len_cv(6),set_len_cv(7));

  f_priors+=neg_log_prior(len_cv_val_20,set_len_cv_20(5),set_len_cv_20(6),set_len_cv_20(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(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_A50_cH1,set_selpar_A50_cH1(5), set_selpar_A50_cH1(6),
set_selpar_A50_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_A50_cH2,set_selpar_A50_cH2(5), set_selpar_A50_cH2(6),
set_selpar_A50_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_A50_cH3,set_selpar_A50_cH3(5), set_selpar_A50_cH3(6),
set_selpar_A50_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_A50_HB1,set_selpar_A50_HB1(5), set_selpar_A50_HB1(6),
set_selpar_A50_HB1(7));

  f_priors+=neg_log_prior(selpar_slope_HB1,set_selpar_slope_HB1(5), set_selpar_slope_HB1(6),
set_selpar_slope_HB1(7));

  f_priors+=neg_log_prior(selpar_A502_HB1,set_selpar_A502_HB1(5), set_selpar_A502_HB1(6),
set_selpar_A502_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_A50_HB2,set_selpar_A50_HB2(5), set_selpar_A50_HB2(6),
set_selpar_A50_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_A502_HB2,set_selpar_A502_HB2(5), set_selpar_A502_HB2(6),
set_selpar_A502_HB2(7));

  f_priors+=neg_log_prior(selpar_slope2_HB2,set_selpar_slope2_HB2(5), set_selpar_slope2_HB2(6),
set_selpar_slope2_HB2(7));

  f_priors+=neg_log_prior(selpar_A50_HB3,set_selpar_A50_HB3(5), set_selpar_A50_HB3(6),
set_selpar_A50_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_A502_HB3,set_selpar_A502_HB3(5), set_selpar_A502_HB3(6),
set_selpar_A502_HB3(7));

  f_priors+=neg_log_prior(selpar_slope2_HB3,set_selpar_slope2_HB3(5), set_selpar_slope2_HB3(6),
set_selpar_slope2_HB3(7));



  f_priors+=neg_log_prior(selpar_A50_GR2,set_selpar_A50_GR2(5), set_selpar_A50_GR2(6),
set_selpar_A50_GR2(7));

  f_priors+=neg_log_prior(selpar_slope_GR2,set_selpar_slope_GR2(5), set_selpar_slope_GR2(6),
set_selpar_slope_GR2(7));

  f_priors+=neg_log_prior(selpar_A502_GR2,set_selpar_A502_GR2(5), set_selpar_A502_GR2(6),
set_selpar_A502_GR2(7));

  f_priors+=neg_log_prior(selpar_slope2_GR2,set_selpar_slope2_GR2(5), set_selpar_slope2_GR2(6),
set_selpar_slope2_GR2(7));



  f_priors+=neg_log_prior(selpar_A50_GR3,set_selpar_A50_GR3(5), set_selpar_A50_GR3(6),
set_selpar_A50_GR3(7));

  f_priors+=neg_log_prior(selpar_slope_GR3,set_selpar_slope_GR3(5), set_selpar_slope_GR3(6),
set_selpar_slope_GR3(7));



  f_priors+=neg_log_prior(selpar_A50_cH2_D,set_selpar_A50_cH2_D(5), set_selpar_A50_cH2_D(6),
set_selpar_A50_cH2_D(7));

  f_priors+=neg_log_prior(selpar_slope_cH2_D,set_selpar_slope_cH2_D(5), set_selpar_slope_cH2_D(6),
set_selpar_slope_cH2_D(7));

  f_priors+=neg_log_prior(selpar_A502_cH2_D,set_selpar_A502_cH2_D(5), set_selpar_A502_cH2_D(6),
set_selpar_A502_cH2_D(7));

  f_priors+=neg_log_prior(selpar_slope2_cH2_D,set_selpar_slope2_cH2_D(5), set_selpar_slope2_cH2_D(6),
set_selpar_slope2_cH2_D(7));
```

```
  f_priors+=neg_log_prior(selpar_A50_cH3_D,set_selpar_A50_cH3_D(5), set_selpar_A50_cH3_D(6),
set_selpar_A50_cH3_D(7));

  f_priors+=neg_log_prior(selpar_slope_cH3_D,set_selpar_slope_cH3_D(5), set_selpar_slope_cH3_D(6),
set_selpar_slope_cH3_D(7));


  f_priors+=neg_log_prior(selpar_A50_HB2_D,set_selpar_A50_HB2_D(5), set_selpar_A50_HB2_D(6),
set_selpar_A50_HB2_D(7));

  f_priors+=neg_log_prior(selpar_slope_HB2_D,set_selpar_slope_HB2_D(5), set_selpar_slope_HB2_D(6),
set_selpar_slope_HB2_D(7));

  f_priors+=neg_log_prior(selpar_A502_HB2_D,set_selpar_A502_HB2_D(5), set_selpar_A502_HB2_D(6),
set_selpar_A502_HB2_D(7));

  f_priors+=neg_log_prior(selpar_slope2_HB2_D,set_selpar_slope2_HB2_D(5), set_selpar_slope2_HB2_D(6),
set_selpar_slope2_HB2_D(7));


  f_priors+=neg_log_prior(selpar_A50_HB3_D,set_selpar_A50_HB3_D(5), set_selpar_A50_HB3_D(6),
set_selpar_A50_HB3_D(7));

  f_priors+=neg_log_prior(selpar_slope_HB3_D,set_selpar_slope_HB3_D(5), set_selpar_slope_HB3_D(6),
set_selpar_slope_HB3_D(7));

  f_priors+=neg_log_prior(selpar_A502_HB3_D,set_selpar_A502_HB3_D(5), set_selpar_A502_HB3_D(6),
set_selpar_A502_HB3_D(7));

  f_priors+=neg_log_prior(selpar_slope2_HB3_D,set_selpar_slope2_HB3_D(5), set_selpar_slope2_HB3_D(6),
set_selpar_slope2_HB3_D(7));


  f_priors+=neg_log_prior(selpar_A50_CVT,set_selpar_A50_CVT(5), set_selpar_A50_CVT(6),
set_selpar_A50_CVT(7));

  f_priors+=neg_log_prior(selpar_slope_CVT,set_selpar_slope_CVT(5), set_selpar_slope_CVT(6),
set_selpar_slope_CVT(7));


  f_priors+=neg_log_prior(log_q_cH,set_log_q_cH(5),set_log_q_cH(6),set_log_q_cH(7));

  f_priors+=neg_log_prior(log_q_HB,set_log_q_HB(5),set_log_q_HB(6),set_log_q_HB(7));

  f_priors+=neg_log_prior(log_q_HB_D,set_log_q_HB_D(5),set_log_q_HB_D(6),set_log_q_HB_D(7));

  f_priors+=neg_log_prior(log_q_CVT,set_log_q_CVT(5),set_log_q_CVT(6),set_log_q_CVT(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(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;

//--------------------------------------------------------------------------------
//Logistic function: 2 parameters
FUNCTION dvar_vector logistic(const dvar_vector& ages, const dvariable& A50, const dvariable& slope)
  //ages=vector of ages, A50=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-A50))); //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& A50, const dvariable&
slope, const dvariable& sigma, const dvariable& joint)
  //ages=vector of ages, A50=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)-A50)));}
```

```
   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& A501, const dvariable&
slope1, const dvariable& A502, const dvariable& slope2)

  //ages=vector of ages, A50=age at 50% selectivity, slope=rate of increase, A502=age at 50% decrease additive to
A501, 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-A501)))),(1.-(1./(1.+mfexp(-1.*slope2*(ages-
(A501+A502)))))) );

  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& A501, const dvariable& slope1,
const dvariable& A502, const dvariable& slope2, const dvariable& satval, const dvariable& joint)

  //ages=vector of ages, A501=age at 50% sel (ascending limb), slope1=rate of increase,A502=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)-A501)));}
  if (double(iage)>joint){Sel_Tmp(iage)=1.0-(1.0-satval)/(1.+mfexp(-1.*slope2*(ages(iage)-A502)));}
 }
 Sel_Tmp=Sel_Tmp/max(Sel_Tmp);
 RETURN_ARRAYS_DECREMENT();
 return Sel_Tmp;


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

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

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

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

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

 RETURN_ARRAYS_INCREMENT();

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

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

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

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

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

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

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

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


 pars_tmp(1)=peak;

 pars_tmp(2)=peak+1.0+(0.99*ages(nages)-peak-1.0)/(1.0+mfexp(-top));

 pars_tmp(3)=mfexp(ascwid);

 pars_tmp(4)=mfexp(deswid);

 pars_tmp(5)=1.0/(1.0+mfexp(-init));
```

```
  pars_tmp(6)=1.0/(1.0+mfexp(-final));

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

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

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

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

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

```
    Recruits_Tmp=((0.8*R0*h*SSB)/(0.2*R0*spr_F0*(1.0-h)+(h-0.2)*SSB));
   break;
   case 2: //Ricker
    Recruits_Tmp=((SSB/spr_F0)*mfexp(h*(1-SSB/(R0*spr_F0))));
   break;
  }
 RETURN_ARRAYS_DECREMENT();
 return Recruits_Tmp;


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


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

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

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

//--------------------------------------------------------------------------------
//Likelihood contribution: multinomial
```

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

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

  RETURN_ARRAYS_INCREMENT();

  dvariable LkvalTmp;

  dvariable small_number=0.00001;

  LkvalTmp=0.0;

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

  {if (nsamp(ii)>=minSS)

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

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

    }

  }

  RETURN_ARRAYS_DECREMENT();

  return LkvalTmp;


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

//Likelihood contribution: multinomial

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

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

  RETURN_ARRAYS_INCREMENT();

  dvariable LkvalTmp;

  dvariable small_number=0.00001;

  LkvalTmp=0.0;

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

  dvar_vector nsamp_wgt=nsamp*wgt_dat;

  //cout<<nsamp_wgt<<endl;
```

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

  {if (nsamp(ii)>=minSS)

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

    }

  }

  RETURN_ARRAYS_DECREMENT();

  return LkvalTmp;


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

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

//Likelihood contribution: priors

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

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

    dvariable LkvalTmp;

    dvariable alpha, beta, ab_iq;

    dvariable big_number=1e10;

    LkvalTmp=0.0;

    // compute generic pdf's

    switch(pdf) {

        case 1: //option to turn off prior

          LkvalTmp=0.0;

          break;

        case 2: // lognormal

          if(prior<=0.0) cout << "YIKES: Don't use a lognormal distn for a negative prior" << endl;

          else if(pred<=0) LkvalTmp=big_number=1e10;

          else {

            if(var<0.0) var=log(1.0+var*var) ;      // convert cv to variance on log scale

            LkvalTmp= 0.5*( square(log(pred/prior))/var + log(var) );
```

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


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

//SDNR: age comp likelihood (assumes fits are done with the robust multinomial function)

FUNCTION dvariable sdnr_multinomial(const double& ncomp, const dvar_vector& ages, const dvar_vector&
nsamp,
                        const dvar_matrix& pred_comp, const dvar_matrix& obs_comp, const dvariable& wgt_dat)

 //ncomp=number of years of data, ages=vector of ages, nsamp=vector of N's,

 //pred_comp=matrix of predicted comps, obs_comp=matrix of observed comps, wgt_dat=likelihood weight for
data source
```

```
 RETURN_ARRAYS_INCREMENT();
 dvariable SdnrTmp;
 dvar_vector o(1,ncomp);
 dvar_vector p(1,ncomp);
 dvar_vector ose(1,ncomp);
 dvar_vector res(1,ncomp);
 SdnrTmp=0.0;
 for (int ii=1; ii<=ncomp; ii++)
 {
  o(ii)=sum(elem_prod(ages,obs_comp(ii)));
  p(ii)=sum(elem_prod(ages,pred_comp(ii)));
  ose(ii)=sqrt((sum(elem_prod(square(ages),pred_comp(ii)))-square(p(ii)))/(nsamp(ii)*wgt_dat));
 }
 res=elem_div((o-p),ose);
 SdnrTmp=sqrt(sum(square(res-(sum(res)/ncomp))/(ncomp-1.0)));
 RETURN_ARRAYS_DECREMENT();
 return SdnrTmp;


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

//SDNR: lognormal likelihood

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

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

 RETURN_ARRAYS_INCREMENT();
 dvariable SdnrTmp;
 dvariable small_number=0.00001;
 dvariable n;
 dvar_vector res(cv.indexmin(),cv.indexmax());
 SdnrTmp=0.0;
```

```
  res=elem_div(log(elem_div(obs+small_number,pred+small_number)),sqrt(log(1+square(cv/wgt_dat))));
  n=cv.indexmax()-cv.indexmin()+1;
  SdnrTmp=sqrt(sum(square(res-(sum(res)/n))/(n-1.0)));
  RETURN_ARRAYS_DECREMENT();
  return SdnrTmp;

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

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

    sdnr_lc_cH=sdnr_multinomial(nyr_cH_lenc, lenbins, nsamp_cH_lenc, pred_cH_lenc, obs_cH_lenc, w_lc_cH);
    sdnr_lc_cH_D=sdnr_multinomial(nyr_cH_D_lenc, lenbins, nsamp_cH_D_lenc, pred_cH_D_lenc,
obs_cH_D_lenc, w_lc_cH_D);
    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_ac_cH=sdnr_multinomial(nyr_cH_agec, agebins_agec, nsamp_cH_agec, pred_cH_agec, obs_cH_agec,
w_ac_cH);

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

    sdnr_ac_CVT=sdnr_multinomial(nyr_CVT_agec, agebins_agec, nsamp_CVT_agec, pred_CVT_agec,
obs_CVT_agec, w_ac_CVT);  //**KIS

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


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

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

    sdnr_I_HB_D=sdnr_lognormal(pred_HB_D_cpue, obs_HB_D_cpue, HB_D_cpue_cv, w_I_HB_D);

    sdnr_I_CVT=sdnr_lognormal(pred_CVT_cpue, obs_CVT_cpue, CVT_cpue_cv, w_I_CVT);  //**KIS




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

    //##  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_20_out(8)=Linf_20; Linf_20_out(1,7)=set_Linf_20;
K_20_out(8)=K_20; K_20_out(1,7)=set_K_20;
t0_20_out(8)=t0_20; t0_20_out(1,7)=set_t0_20;
len_cv_val_20_out(8)=len_cv_val_20; len_cv_val_20_out(1,7)=set_len_cv_20;

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_A50_cH1_out(8)=selpar_A50_cH1; selpar_A50_cH1_out(1,7)=set_selpar_A50_cH1;
selpar_slope_cH1_out(8)=selpar_slope_cH1; selpar_slope_cH1_out(1,7)=set_selpar_slope_cH1;
selpar_A50_cH2_out(8)=selpar_A50_cH2; selpar_A50_cH2_out(1,7)=set_selpar_A50_cH2;
selpar_slope_cH2_out(8)=selpar_slope_cH2; selpar_slope_cH2_out(1,7)=set_selpar_slope_cH2;
selpar_A50_cH3_out(8)=selpar_A50_cH3; selpar_A50_cH3_out(1,7)=set_selpar_A50_cH3;
selpar_slope_cH3_out(8)=selpar_slope_cH3; selpar_slope_cH3_out(1,7)=set_selpar_slope_cH3;

selpar_A50_HB1_out(8)=selpar_A50_HB1; selpar_A50_HB1_out(1,7)=set_selpar_A50_HB1;
selpar_slope_HB1_out(8)=selpar_slope_HB1; selpar_slope_HB1_out(1,7)=set_selpar_slope_HB1;
selpar_A502_HB1_out(8)=selpar_A502_HB1; selpar_A502_HB1_out(1,7)=set_selpar_A502_HB1;
selpar_slope2_HB1_out(8)=selpar_slope2_HB1; selpar_slope2_HB1_out(1,7)=set_selpar_slope2_HB1;
selpar_A50_HB2_out(8)=selpar_A50_HB2; selpar_A50_HB2_out(1,7)=set_selpar_A50_HB2;
selpar_slope_HB2_out(8)=selpar_slope_HB2; selpar_slope_HB2_out(1,7)=set_selpar_slope_HB2;
selpar_A502_HB2_out(8)=selpar_A502_HB2; selpar_A502_HB2_out(1,7)=set_selpar_A502_HB2;
selpar_slope2_HB2_out(8)=selpar_slope2_HB2; selpar_slope2_HB2_out(1,7)=set_selpar_slope2_HB2;
selpar_A50_HB3_out(8)=selpar_A50_HB3; selpar_A50_HB3_out(1,7)=set_selpar_A50_HB3;
selpar_slope_HB3_out(8)=selpar_slope_HB3; selpar_slope_HB3_out(1,7)=set_selpar_slope_HB3;
selpar_A502_HB3_out(8)=selpar_A502_HB3; selpar_A502_HB3_out(1,7)=set_selpar_A502_HB3;
```

```
selpar_slope2_HB3_out(8)=selpar_slope2_HB3; selpar_slope2_HB3_out(1,7)=set_selpar_slope2_HB3;


selpar_A50_HB2_D_out(8)=selpar_A50_HB2_D; selpar_A50_HB2_D_out(1,7)=set_selpar_A50_HB2_D;

selpar_slope_HB2_D_out(8)=selpar_slope_HB2_D; selpar_slope_HB2_D_out(1,7)=set_selpar_slope_HB2_D;

selpar_A502_HB2_D_out(8)=selpar_A502_HB2_D;
selpar_A502_HB2_D_out(1,7)=set_selpar_A502_HB2_D;

selpar_slope2_HB2_D_out(8)=selpar_slope2_HB2_D;
selpar_slope2_HB2_D_out(1,7)=set_selpar_slope2_HB2_D;


        selpar_A50_HB3_D_out(8)=selpar_A50_HB3_D;
selpar_A50_HB3_D_out(1,7)=set_selpar_A50_HB3_D;

        selpar_slope_HB3_D_out(8)=selpar_slope_HB3_D;
selpar_slope_HB3_D_out(1,7)=set_selpar_slope_HB3_D;

        selpar_A502_HB3_D_out(8)=selpar_A502_HB3_D;
selpar_A502_HB3_D_out(1,7)=set_selpar_A502_HB3_D;

        selpar_slope2_HB3_D_out(8)=selpar_slope2_HB3_D;
selpar_slope2_HB3_D_out(1,7)=set_selpar_slope2_HB3_D;


        selpar_A50_cH2_D_out(8)=selpar_A50_cH2_D; selpar_A50_cH2_D_out(1,7)=set_selpar_A50_cH2_D;

        selpar_slope_cH2_D_out(8)=selpar_slope_cH2_D;
selpar_slope_cH2_D_out(1,7)=set_selpar_slope_cH2_D;

        selpar_A502_cH2_D_out(8)=selpar_A502_cH2_D;
selpar_A502_cH2_D_out(1,7)=set_selpar_A502_cH2_D;

        selpar_slope2_cH2_D_out(8)=selpar_slope2_cH2_D;
selpar_slope2_cH2_D_out(1,7)=set_selpar_slope2_cH2_D;


        selpar_A50_cH3_D_out(8)=selpar_A50_cH3_D; selpar_A50_cH3_D_out(1,7)=set_selpar_A50_cH3_D;

        selpar_slope_cH3_D_out(8)=selpar_slope_cH3_D;
selpar_slope_cH3_D_out(1,7)=set_selpar_slope_cH3_D;


        selpar_A50_GR2_out(8)=selpar_A50_GR2; selpar_A50_GR2_out(1,7)=set_selpar_A50_GR2;

selpar_slope_GR2_out(8)=selpar_slope_GR2; selpar_slope_GR2_out(1,7)=set_selpar_slope_GR2;

selpar_A502_GR2_out(8)=selpar_A502_GR2; selpar_A502_GR2_out(1,7)=set_selpar_A502_GR2;

selpar_slope2_GR2_out(8)=selpar_slope2_GR2; selpar_slope2_GR2_out(1,7)=set_selpar_slope2_GR2;
```

```
selpar_A50_GR3_out(8)=selpar_A50_GR3; selpar_A50_GR3_out(1,7)=set_selpar_A50_GR3;
selpar_slope_GR3_out(8)=selpar_slope_GR3; selpar_slope_GR3_out(1,7)=set_selpar_slope_GR3;
//selpar_A502_GR3_out(8)=selpar_A502_GR3; selpar_A502_GR3_out(1,7)=set_selpar_A502_GR3;
//selpar_slope2_GR3_out(8)=selpar_slope2_GR3; selpar_slope2_GR3_out(1,7)=set_selpar_slope2_GR3;

selpar_A50_CVT_out(8)=selpar_A50_CVT; selpar_A50_CVT_out(1,7)=set_selpar_A50_CVT;
selpar_slope_CVT_out(8)=selpar_slope_CVT; selpar_slope_CVT_out(1,7)=set_selpar_slope_CVT;

log_q_cH_out(8)=log_q_cH; log_q_cH_out(1,7)=set_log_q_cH;
log_q_HB_out(8)=log_q_HB; log_q_HB_out(1,7)=set_log_q_HB;
log_q_HB_D_out(8)=log_q_HB_D; log_q_HB_D_out(1,7)=set_log_q_HB_D;
log_q_CVT_out(8)=log_q_CVT; log_q_CVT_out(1,7)=set_log_q_CVT;

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_cH_D_out(8)=log_avg_F_cH_D; log_avg_F_cH_D_out(1,7)=set_log_avg_F_cH_D;
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_cH_D_out(styr_cH_D,endyr_cH_D)=log_F_dev_cH_D;
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 "RS_make_Robjectbase.cxx"   // write the R-compatible report

  } //endl last phase loop
```

ADMB Dat file for the BAM

```
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>
##
##  Data Input File
##  RS for SEDAR 41
##
##--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>--><>

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

#Starting and ending year of model
1950
2014
#Starting year to estimate recruitment deviation from S-R curve
1978
```

#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

1978

2013

#Ending year for selectivity blocks

1983

1991

2009

#Number of ages in population model(16 classes are 1,...,N+) //assumes last age is plus group

20

#Vector of agebins, last is a plus group

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0

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

13

#Number of ages used to match HB age comps only

10.0

#Vector of agebins, last is a plus group

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0

#Vector of agebins for HB only

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

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

27 #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

210 240 270 300 330 360 390 420 450 480 510 540 570
600 630 660 690 720 750 780 810 840 870 900 930
960 990

#Max value of F used in spr and msy calculations

1.0

#Number of iterations in spr and msy calculations

10001

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

3

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

-1.0

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

##-- BAM DATA SECTION: observed data section

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

###Commercial Handline
###

##Comm handline Index

##Starting and ending years of CPUE index

1993

2009

##Observed CPUE and CVs

1.09 0.89 0.89 0.61 0.59 0.66 0.8 0.74 1.27 1.38 1.04 1.42 1.19 0.6 0.67 1.22 1.94

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

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

###Starting and ending years for landings time series

1950

2014

##commercial handline landings vector (1000 lb whole weight) and assumed CVs

368.657 499.765 385.930 398.279 593.207 493.315 483.907 867.291 612.508 657.736 671.075 796.374 645.983
488.789 537.589 558.108 554.506 725.503 865.520 538.190 513.023 457.393 406.641 296.560 478.352
600.790 571.504 596.339 594.356 420.936 385.485 378.759 308.445 316.818 253.431 250.824 219.440
191.701 173.689 266.942 226.542 143.546 104.374 220.153 195.319 177.312 138.671 110.595 89.602
93.595 104.165 196.697 187.967 138.342 172.083 129.700 86.382 114.973 252.146 362.386 6.448
0.568 8.142 31.600 65.443

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

#Number and vector of years of commercial handline length compositions

9

1984 1985 1986 1987 1988 1989 1990 1991 1992 #1993 1994 1995
1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007
2008 2009 2010 2012 2013 2014

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

125.0 139.0 94.0 89.0 84.0 88.0 63.0 106.0 82.0 #116.0 102.0 127.0 157.0 105.0 121.0 167.0 149.0 196.0 144.0
155.0 138.0 133.0 133.0 185.0 172.0 261.0 3.0 40.0 92.0 61.0

2089.0 2162.0 933.0 748.0 539.0 828.0 608.0 527.0 366.0 #728.0 773.0 1100.0 872.0 460.0 467.0 1026.0 945.0
1894.0 1009.0 1385.0 880.0 615.0 436.0 588.0 673.0 2548.0 67.0 139.0 458.0 367.0

#commercial handline length comps (3 cm length bins)

0.0000 0.0000 0.0000 0.0007 0.0089 0.0295 0.0467 0.2647 0.1739 0.1362 0.1137 0.0638 0.0626
0.0330 0.0078 0.0043 0.0039 0.0160 0.0014 0.0006 0.0019 0.0172 0.0058 0.0056 0.0011
0.0006 0.0002

0.0000 0.0000 0.0007 0.0016 0.0040 0.0314 0.0930 0.1819 0.2000 0.1165 0.0916 0.0636 0.0770
0.0356 0.0208 0.0094 0.0050 0.0078 0.0036 0.0062 0.0034 0.0098 0.0129 0.0101 0.0097
0.0043 0.0000

0.0000 0.0000 0.0000 0.0027 0.0155 0.0385 0.0407 0.0548 0.1074 0.1695 0.0956 0.0907 0.0552
0.0771 0.0557 0.0390 0.0310 0.0216 0.0117 0.0058 0.0079 0.0102 0.0107 0.0338 0.0088
0.0063 0.0099

0.0000 0.0000 0.0000 0.0107 0.0608 0.1254 0.0808 0.1244 0.1098 0.0732 0.0348 0.0382 0.0475
0.0678 0.0598 0.0481 0.0228 0.0206 0.0102 0.0033 0.0062 0.0017 0.0075 0.0167 0.0124
0.0065 0.0107

0.0000 0.0000 0.0030 0.0056 0.0256 0.0795 0.1104 0.1103 0.0802 0.0951 0.0749 0.0746 0.0720
0.0505 0.0488 0.0185 0.0189 0.0166 0.0250 0.0039 0.0157 0.0166 0.0167 0.0164 0.0131
0.0084 0.0000

0.0000 0.0006 0.0000 0.0035 0.0041 0.0349 0.0788 0.1245 0.1003 0.0892 0.1208 0.0639 0.0868
0.0592 0.0306 0.0312 0.0114 0.0176 0.0182 0.0207 0.0089 0.0264 0.0345 0.0193 0.0084
0.0050 0.0012

0.0000 0.0000 0.0000 0.0000 0.0130 0.0849 0.1793 0.0957 0.0711 0.0645 0.0910 0.0477 0.0511
0.0677 0.0470 0.0398 0.0320 0.0172 0.0307 0.0123 0.0047 0.0231 0.0027 0.0104 0.0088
0.0054 0.0000

0.0000 0.0000 0.0000 0.0221 0.0938 0.0743 0.0669 0.0744 0.0686 0.0493 0.0828 0.0429 0.0522
0.0222 0.0590 0.0309 0.0286 0.0365 0.0180 0.0255 0.0296 0.0313 0.0422 0.0293 0.0125
0.0039 0.0030

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0027 0.0095 0.0383 0.1162 0.0747 0.0731
0.0786 0.0849 0.0955 0.0886 0.0507 0.0524 0.0486 0.0418 0.0417 0.0379 0.0240 0.0332
0.0044 0.0031

#0.0000 0.0000 0.0000 0.0000 0.0010 0.0000 0.0029 0.0097 0.0141 0.0365 0.1414 0.1560 0.1170
0.0746 0.0620 0.0327 0.0382 0.0532 0.0365 0.0416 0.0394 0.0353 0.0326 0.0402 0.0216
0.0084 0.0051

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0258 0.1347 0.1762 0.1858
0.1250 0.1190 0.0691 0.0383 0.0184 0.0146 0.0150 0.0093 0.0113 0.0153 0.0171 0.0206
0.0045 0.0000

#0.0000 0.0000 0.0000 0.0007 0.0007 0.0036 0.0017 0.0052 0.0099 0.0234 0.0960 0.1412 0.1180
0.1204 0.0982 0.0649 0.1107 0.0597 0.0376 0.0163 0.0142 0.0204 0.0217 0.0149 0.0156
0.0026 0.0023

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0012 0.0021 0.0010 0.0000 0.0065 0.0586 0.1258 0.1271
0.1245 0.1381 0.0688 0.0711 0.0908 0.0706 0.0472 0.0199 0.0138 0.0155 0.0093 0.0072
0.0000 0.0011

#0.0000 0.0000 0.0000 0.0000 0.0016 0.0049 0.0045 0.0041 0.0019 0.0050 0.0478 0.0561 0.0701
0.1235 0.1286 0.1016 0.0992 0.0983 0.0839 0.0497 0.0535 0.0236 0.0125 0.0061 0.0115
0.0072 0.0048

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0020 0.0031 0.0028 0.0127 0.1144 0.1163 0.1254 0.1203
0.0725 0.0455 0.0341 0.0680 0.0722 0.0643 0.0441 0.0211 0.0288 0.0138 0.0127 0.0200
0.0059 0.0000

#0.0000 0.0000 0.0000 0.0004 0.0009 0.0071 0.0198 0.0154 0.0070 0.0527 0.1385 0.1347 0.1037
0.1351 0.1098 0.0659 0.0385 0.0204 0.0169 0.0281 0.0408 0.0107 0.0145 0.0258 0.0096
0.0012 0.0025

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0015 0.0618 0.2155 0.1577 0.1139
0.0901 0.0558 0.0385 0.0513 0.0340 0.0303 0.0182 0.0402 0.0315 0.0308 0.0141 0.0126
0.0000 0.0022

#0.0000 0.0000 0.0005 0.0287 0.0362 0.0184 0.0168 0.0076 0.0078 0.0616 0.1737 0.1874 0.1452
0.1086 0.0543 0.0379 0.0310 0.0177 0.0154 0.0134 0.0101 0.0099 0.0073 0.0034 0.0053
0.0009 0.0009

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0014 0.0387 0.1648 0.1818 0.1487
0.1196 0.1087 0.0623 0.0647 0.0499 0.0106 0.0132 0.0081 0.0128 0.0045 0.0000 0.0051
0.0027 0.0025

#0.0000 0.0000 0.0000 0.0090 0.0142 0.0082 0.0141 0.0177 0.0227 0.0213 0.0643 0.1015 0.1332
0.1233 0.1291 0.0771 0.0861 0.0594 0.0545 0.0268 0.0070 0.0076 0.0118 0.0047 0.0016
0.0049 0.0000

#0.0000 0.0000 0.0000 0.0106 0.0153 0.0203 0.0185 0.0095 0.0049 0.0181 0.0514 0.0801 0.1094
0.1064 0.1142 0.0862 0.1129 0.1086 0.0683 0.0365 0.0091 0.0075 0.0070 0.0015 0.0004
0.0032 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0058 0.0000 0.0254 0.1131 0.0638 0.0841
0.1073 0.0689 0.0787 0.0826 0.0933 0.1197 0.0727 0.0369 0.0239 0.0098 0.0066 0.0061
0.0004 0.0007

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0032 0.0027 0.0034 0.0175 0.0489 0.1163 0.1021
0.0804 0.0984 0.0522 0.0757 0.0909 0.0765 0.0662 0.0778 0.0386 0.0238 0.0185 0.0070
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0012 0.0032 0.0042 0.0039 0.0041 0.0625 0.1923 0.1196 0.0366
0.0370 0.0442 0.0487 0.0626 0.0656 0.0906 0.0627 0.0556 0.0638 0.0211 0.0189 0.0016
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0024 0.0007 0.0012 0.0775 0.2815 0.2337 0.1385
0.0746 0.0281 0.0124 0.0079 0.0196 0.0295 0.0384 0.0141 0.0266 0.0092 0.0032 0.0008
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0022 0.0046 0.0295 0.1144 0.1639 0.1450
0.1476 0.1116 0.0530 0.0559 0.0342 0.0205 0.0319 0.0321 0.0255 0.0208 0.0052 0.0016
0.0005 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.1209 0.5610 0.1495
0.0992 0.0411 0.0187 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0095 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0185 0.0000 0.0000 0.0000 0.0333 0.0265 0.0477 0.0000 0.0072 0.0518 0.0280
0.0564 0.1015 0.0987 0.1078 0.1476 0.1473 0.0879 0.0062 0.0090 0.0247 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0030 0.0000 0.0000 0.0071 0.0112 0.0087 0.0253 0.0225 0.0705 0.0902 0.0889 0.0621
0.0493 0.0883 0.0716 0.0645 0.0900 0.0737 0.0787 0.0397 0.0402 0.0103 0.0043 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0049 0.0165 0.0033 0.0460 0.0338 0.0255 0.0205 0.0358 0.0431 0.0622 0.0567
0.0809 0.0899 0.0728 0.0919 0.0946 0.0830 0.0548 0.0332 0.0308 0.0130 0.0066 0.0000
0.0000 0.0000

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

19

#24

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

1990 1992 1994 1996 1997 1998 1999 2000 2001 2003 2004 2005 2006
2007 2008 2009 2012 2013 2014

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

#7.0 11.0 11.0 8.0 14.0 2.0 48.0 45.0 14.0 15.0 28.0 23.0 5.0 10.0 25.0 53.0 84.0 132.0 158.0 263.0 1.0 39.0 109.0 64.0

#32.0 29.0 38.0 12.0 20.0 16.0 204.0 174.0 50.0 164.0 288.0 115.0 30.0 59.0 78.0 135.0 229.0 267.0 389.0 2432.0 30.0 148.0 722.0 465.0

11.0 11.0 14.0 48.0 45.0 14.0 15.0 28.0 23.0 10.0 25.0 53.0 84.0 132.0 158.0 263.0 39.0 109.0 64.0

29.0 38.0 20.0 204.0 174.0 50.0 164.0 288.0 115.0 59.0 78.0 135.0 229.0 267.0 389.0 2432.0 148.0 722.0 465.0

#commercial handline age comps

#0.0000 0.1111 0.3298 0.5176 0.0415 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.1298 0.4010 0.4691 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.5567 0.2080 0.0570 0.0780 0.0234 0.0000 0.0769 0.0000 0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.6860 0.3045 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0095

0.0000 0.0000 0.0000 0.0000 0.3998 0.3175 0.2127 0.0610 0.0000 0.0000 0.0000 0.0000 0.0090

#0.0000 0.3844 0.2330 0.2910 0.0000 0.0000 0.0000 0.0634 0.0281 0.0000 0.0000 0.0000 0.0000

0.0000 0.0037 0.1931 0.1034 0.2015 0.2155 0.1546 0.0452 0.0097 0.0184 0.0079 0.0135 0.0334

0.0000 0.0431 0.0762 0.4328 0.1112 0.1697 0.0593 0.0362 0.0356 0.0129 0.0000 0.0000 0.0231

0.0000 0.0593 0.4308 0.0714 0.2170 0.0943 0.0275 0.0211 0.0236 0.0143 0.0061 0.0092 0.0254

0.0000 0.0493 0.2204 0.5495 0.0213 0.0280 0.0469 0.0375 0.0085 0.0031 0.0017 0.0029 0.0310

0.0000 0.0140 0.4217 0.2879 0.0941 0.0167 0.0297 0.0800 0.0065 0.0071 0.0054 0.0000 0.0368

0.0000 0.1155 0.4626 0.3102 0.0178 0.0184 0.0062 0.0143 0.0030 0.0203 0.0069 0.0060 0.0190

#0.0000 0.0000 0.5104 0.3473 0.0460 0.0880 0.0083 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0789 0.1617 0.3619 0.2733 0.0700 0.0000 0.0000 0.0182 0.0100 0.0000 0.0063 0.0196

0.0000 0.0543 0.2660 0.4558 0.1497 0.0501 0.0138 0.0103 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0038 0.3094 0.2445 0.2389 0.1121 0.0478 0.0182 0.0118 0.0000 0.0000 0.0000 0.0134

0.0000 0.0052 0.0990 0.4711 0.0708 0.0766 0.1669 0.0114 0.0060 0.0173 0.0158 0.0000 0.0599

0.0000 0.4304 0.0172 0.0459 0.1263 0.1020 0.1222 0.0793 0.0350 0.0191 0.0000 0.0008 0.0218

0.0000 0.0341 0.8734 0.0373 0.0027 0.0238 0.0044 0.0066 0.0112 0.0026 0.0010 0.0003 0.0028

0.0000 0.0097 0.3498 0.4944 0.0083 0.0080 0.0308 0.0211 0.0220 0.0197 0.0138 0.0072 0.0152

#0.0000 0.2602 0.6169 0.1077 0.0000 0.0000 0.0000 0.0152 0.0000 0.0000 0.0000 0.0000 0.0000

0.0198 0.0921 0.0915 0.0734 0.5369 0.1090 0.0493 0.0000 0.0000 0.0051 0.0086 0.0000 0.0142

0.0063 0.0630 0.2660 0.2852 0.0608 0.1504 0.0831 0.0478 0.0073 0.0000 0.0017 0.0017 0.0265

0.0533 0.1145 0.1095 0.2492 0.1259 0.0238 0.1768 0.0637 0.0494 0.0056 0.0019 0.0112 0.0153

#Discards

#Starting and ending years of discards time series, respectively

1992

2014

2006 #year of hook reg

#Observed discards (1000s) and assumed CVs

19.603 16.725 21.134 21.068 20.727 22.392 16.171 13.641 14.552 15.141 29.848 8.372 2.425
10.177 4.817 13.778 12.553 14.466 17.438 40.107 19.214 19.302 27.008

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 the Comm fleet (pooled years 07-09 and 10-13)

2

2009 2013

#sample size of Comm length comp data by year (first row Ntrips, second row Nfish)

13 13

144 54

#Comm discard length comp samples (year, 3cm lengthbin) ***unweighted***

0.0000 0.0000 0.0000 0.0069 0.0278 0.1736 0.2083 0.2847 0.2083 0.0764 0.0069 0.0000 0.0000
0.0000 0.0069 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0185 0.0556 0.0556 0.0370 0.1111 0.0926
0.1852 0.0370 0.0926 0.0926 0.0926 0.1296 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

##########headboat##########

#Headboat Index

#Starting and ending years of CPUE index

1976

2009

#Observed CPUE and CVs

2.37 2.16 2.13 2.23 1.45 2.95 1.2 1.64 1.42 2.07 0.48 0.58 0.56
0.9 0.87 0.69 0.08 0.16 0.26 0.28 0.25 0.27 0.24 0.29 0.41
0.76 0.88 0.52 0.76 0.76 0.43 0.44 1.71 1.81

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 0.2 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

#0.05 0.08 0.03 0.05 0.05 0.04 0.05 0.05 0.03 0.05 0.07 0.05 0.06
0.05 0.06 0.04 0.1 0.08 0.05 0.06 0.06 0.09 0.08 0.05 0.05
0.07 0.05 0.05 0.04 0.04 0.05 0.08 0.05 0.03

#HB landings

#Starting and ending years of landings time series, respectively

1955

2014

#Observed Headboat landings (1000s of fish) and assumed CVs

#36.536 39.899 43.263 46.626 49.989 53.353 58.184 63.015 67.847 72.678 77.510 77.964 78.418
78.872 79.326 79.780 87.665 95.549 103.434 111.319119.204 120.549 121.894 123.239 124.584
125.929 36.031 19.553 30.698 31.146 50.336 16.625 24.996 36.527 23.453 20.919 13.857
5.301 7.347 8.225 8.826 5.543 5.770 4.741 6.836 8.437 12.028 12.931 5.706
10.842 8.907 5.945 6.889 18.943 21.507 0.477 1.359 2.127 1.520 5.904

12.501 13.652 14.803 15.953 17.104 18.255 19.908 21.561 23.214 24.867 26.520 26.676 26.831
26.986 27.142 27.297 29.995 32.693 35.391 38.088 40.786 41.246 41.707 42.167 42.627
43.087 36.031 19.553 30.698 31.146 50.336 16.625 24.996 36.527 23.453 20.919 13.857 5.301 7.347
8.225 8.826 5.543 5.770 4.741 6.836 8.437 12.028 12.931 5.706 10.842 8.907 5.945 6.889 18.943 21.507 0.477
1.359 2.127 1.520 5.904

0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 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 HB fleet

#35

#1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990
1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
2003 2004 2005 2006 2007 2008 2009 2012 2013 2014

#sample size of HB length comp data by year (first row observed Ntrips, second row Nfish)

#208.0 80.0 73.0 183.0 154.0 253.0 314.0 298.0 190.0 158.0 116.0 157.0 137.0
64.0 49.0 96.0 57.0 74.0 29.0 33.0 78.0 73.0 59.0 103.0 142.0
145.0 102.0 92.0 91.0 55.0 81.0 166.0 16.0 31.0 42.0

#740.0 230.0 234.0 652.0 457.0 1006.0 1321.0 1191.0 435.0 306.0 204.0 365.0 367.0
152.0 45.0 203.0 120.0 147.0 55.0 57.0 149.0 140.0 107.0 239.0 341.0
329.0 290.0 189.0 159.0 153.0 435.0 738.0 132.0 177.0 291.0

#HB length composition samples (year,lengthbin 3 cm)

#0.0035 0.0158 0.0310 0.0589 0.0899 0.1311 0.1832 0.1302 0.0530 0.0383 0.0566 0.0507 0.0311
0.0133 0.0148 0.0069 0.0189 0.0105 0.0193 0.0065 0.0069 0.0090 0.0085 0.0045 0.0040
0.0025 0.0010

#0.0000 0.0043 0.0000 0.0130 0.0826 0.1696 0.2130 0.1348 0.0348 0.0348 0.0217 0.0217 0.0174
0.0000 0.0130 0.0087 0.0435 0.0478 0.0522 0.0348 0.0217 0.0217 0.0043 0.0043 0.0000
0.0000 0.0000

#0.0000 0.0171 0.0171 0.0598 0.1325 0.1667 0.0812 0.1624 0.0855 0.0855 0.0385 0.0171 0.0128
0.0085 0.0128 0.0128 0.0085 0.0043 0.0085 0.0085 0.0000 0.0043 0.0043 0.0043 0.0214
0.0256 0.0000

#0.0000 0.0046 0.0046 0.0230 0.0721 0.1994 0.1902 0.1887 0.1166 0.0537 0.0353 0.0107 0.0061
0.0107 0.0046 0.0061 0.0077 0.0107 0.0077 0.0107 0.0107 0.0031 0.0077 0.0077 0.0031
0.0015 0.0031

#0.0000 0.0021 0.0104 0.0230 0.0794 0.0986 0.0660 0.0856 0.1107 0.1232 0.0986 0.0777 0.0497
0.0289 0.0130 0.0109 0.0184 0.0042 0.0146 0.0084 0.0096 0.0251 0.0176 0.0121 0.0034
0.0088 0.0000

#0.0000 0.0080 0.0399 0.0787 0.1954 0.2293 0.1645 0.0867 0.0319 0.0325 0.0188 0.0146 0.0088
0.0118 0.0118 0.0079 0.0078 0.0068 0.0059 0.0078 0.0088 0.0039 0.0057 0.0096 0.0020
0.0000 0.0010

#0.0008 0.0071 0.0131 0.0546 0.1388 0.1911 0.1710 0.1622 0.0924 0.0527 0.0291 0.0206 0.0082
0.0074 0.0087 0.0043 0.0047 0.0045 0.0008 0.0037 0.0047 0.0013 0.0066 0.0058 0.0058
0.0000 0.0000

#0.0000 0.0041 0.0066 0.0448 0.1252 0.1826 0.1910 0.1465 0.1001 0.0692 0.0422 0.0261 0.0170
0.0141 0.0069 0.0016 0.0026 0.0019 0.0008 0.0010 0.0035 0.0010 0.0035 0.0024 0.0018
0.0018 0.0018

#0.0025 0.0025 0.0175 0.0752 0.1128 0.0952 0.0991 0.1289 0.0902 0.0629 0.0747 0.0797 0.0476
0.0490 0.0184 0.0120 0.0025 0.0075 0.0025 0.0050 0.0000 0.0000 0.0000 0.0050 0.0078
0.0014 0.0000

#0.0000 0.0209 0.0349 0.0497 0.1044 0.1239 0.1701 0.1415 0.0693 0.0660 0.0358 0.0394 0.0436
0.0239 0.0196 0.0183 0.0190 0.0126 0.0000 0.0000 0.0000 0.0000 0.0070 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0121 0.0099 0.0198 0.0859 0.1362 0.1880 0.1639 0.0686 0.0639 0.0600 0.0191 0.0198
0.0297 0.0236 0.0121 0.0259 0.0137 0.0099 0.0243 0.0000 0.0000 0.0000 0.0076 0.0000
0.0061 0.0000

#0.0000 0.0100 0.0025 0.0426 0.1103 0.2256 0.1103 0.1209 0.0964 0.0370 0.0570 0.0651 0.0454
0.0241 0.0075 0.0025 0.0094 0.0025 0.0110 0.0025 0.0050 0.0025 0.0075 0.0000 0.0025
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0082 0.0738 0.1256 0.2018 0.1582 0.1037 0.1006 0.0679 0.0325 0.0271
0.0299 0.0380 0.0081 0.0027 0.0027 0.0027 0.0000 0.0000 0.0027 0.0109 0.0027 0.0000
0.0000 0.0000

#0.0080 0.0000 0.0325 0.0429 0.0896 0.1788 0.0773 0.1240 0.1324 0.0731 0.0689 0.0283 0.0325
0.0203 0.0122 0.0161 0.0142 0.0122 0.0080 0.0000 0.0000 0.0061 0.0000 0.0161 0.0061
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0222 0.0000 0.0222 0.0000 0.0000 0.0000 0.1333 0.4222 0.1333 0.0222
0.0889 0.0667 0.0000 0.0444 0.0444 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0048 0.0095 0.0388 0.2027 0.2606 0.2075
0.0877 0.0394 0.0489 0.0203 0.0191 0.0095 0.0149 0.0101 0.0107 0.0000 0.0000 0.0101
0.0054 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0122 0.0878 0.2439 0.2299
0.1429 0.0395 0.0561 0.0895 0.0122 0.0378 0.0316 0.0000 0.0000 0.0061 0.0000 0.0000
0.0105 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0301 0.1094 0.1688 0.1653
0.1743 0.0848 0.0641 0.0641 0.0598 0.0297 0.0195 0.0051 0.0098 0.0051 0.0051 0.0000
0.0051 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.2364 0.1091 0.2727
0.1455 0.0909 0.0364 0.0364 0.0364 0.0000 0.0182 0.0000 0.0000 0.0000 0.0182 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0526 0.0175 0.0000 0.0000 0.0000 0.0351 0.1579 0.0877 0.0702
0.0877 0.2281 0.1228 0.0526 0.0175 0.0175 0.0000 0.0175 0.0000 0.0175 0.0175 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0134 0.0201 0.0000 0.0134 0.0134 0.0067 0.0067 0.0067 0.1075 0.2686 0.2953
0.1209 0.0470 0.0067 0.0201 0.0067 0.0134 0.0067 0.0134 0.0067 0.0067 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0071 0.0000 0.0143 0.1357 0.2143 0.2286
0.1857 0.0857 0.0286 0.0357 0.0357 0.0143 0.0071 0.0000 0.0000 0.0071 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0280 0.2243 0.1682 0.2430
0.1682 0.1215 0.0280 0.0093 0.0093 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0042 0.0000 0.0042 0.0293 0.1004 0.2050 0.2301
0.1967 0.0837 0.0628 0.0251 0.0251 0.0042 0.0042 0.0126 0.0000 0.0000 0.0000 0.0084
0.0042 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0059 0.0000 0.0000 0.0029 0.0059 0.0176 0.1730 0.2141 0.1701
0.1877 0.0968 0.0352 0.0264 0.0293 0.0147 0.0117 0.0029 0.0000 0.0059 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0093 0.1579 0.1931 0.1783
0.0956 0.1024 0.0974 0.0765 0.0320 0.0116 0.0195 0.0046 0.0046 0.0023 0.0125 0.0000
0.0023 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0034 0.0000 0.0138 0.0966 0.1483 0.3276
0.2069 0.0724 0.0414 0.0310 0.0207 0.0069 0.0172 0.0000 0.0034 0.0034 0.0034 0.0000
0.0034 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0053 0.0159 0.1164 0.1958 0.2487
0.1429 0.0529 0.0688 0.0476 0.0265 0.0476 0.0106 0.0053 0.0159 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0566 0.1572 0.2767
0.2642 0.0881 0.0314 0.0314 0.0189 0.0126 0.0000 0.0377 0.0126 0.0063 0.0000 0.0063
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0131 0.2484 0.3399 0.1634
0.1046 0.0327 0.0131 0.0261 0.0065 0.0065 0.0000 0.0131 0.0131 0.0065 0.0065 0.0000
0.0065 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0069 0.2414 0.2897 0.2253
0.1241 0.0437 0.0161 0.0046 0.0046 0.0115 0.0046 0.0138 0.0069 0.0023 0.0000 0.0023
0.0000 0.0023

#0.0000 0.0000 0.0000 0.0014 0.0000 0.0027 0.0000 0.0014 0.0068 0.0691 0.3211 0.2100 0.1220
0.0840 0.0610 0.0366 0.0271 0.0136 0.0054 0.0095 0.0068 0.0068 0.0081 0.0054 0.0014
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0395 0.1338 0.0920 0.0761 0.1279 0.0790 0.0489 0.0283 0.0472
0.0489 0.0772 0.0584 0.0336 0.0696 0.0301 0.0094 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0067 0.0013 0.0000 0.0347 0.0306 0.0239 0.0360 0.0787 0.0972 0.1284 0.0468
0.0841 0.0468 0.0841 0.0736 0.0226 0.1029 0.0548 0.0134 0.0067 0.0201 0.0000 0.0000
0.0000 0.0067

#0.0000 0.0323 0.0122 0.0526 0.0162 0.0284 0.1216 0.1297 0.1577 0.0689 0.0365 0.0283 0.0243
0.0547 0.0242 0.0121 0.0222 0.0506 0.0344 0.0344 0.0324 0.0162 0.0081 0.0020 0.0000
0.0000 0.0000

#Number and vector of years of age compositions for headboat fleet

#23

20

#1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1989 1990 1991
1998 2004 2005 2006 2007 2008 2009 2012 2013 2014

1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1990 1991 2005
2006 2007 2008 2009 2012 2013 2014

#sample sizes of age comps by year (first row observed Ntrips, second row Nfish)

#80.0 31.0 30.0 141.0 55.0 167.0 166.0 160.0 97.0 60.0 9.0 23.0 13.0
2.0 8.0 22.0 49.0 34.0 47.0 241.0 40.0 35.0 49.0

#275.0 46.0 87.0 405.0 131.0 741.0 581.0 504.0 184.0 86.0 49.0 33.0 21.0
21.0 27.0 60.0 150.0 71.0 133.0 1239.0 604.0 242.0 364.0

80.0 31.0 30.0 141.0 55.0 167.0 166.0 160.0 97.0 60.0 23.0 13.0 22.0
49.0 34.0 47.0 241.0 40.0 35.0 49.0

275.0 46.0 87.0 405.0 131.0 741.0 581.0 504.0 184.0 86.0 33.0 21.0 60.0
150.0 71.0 133.0 1239.0 604.0 242.0 364.0

#age composition samples (year,age) from headboat fleet

0.0257 0.3790 0.5195 0.0239 0.0353 0.0048 0.0086 0.0000 0.0000 0.0032

0.0000 0.7157 0.0928 0.0470 0.0718 0.0319 0.0408 0.0000 0.0000 0.0000

0.1229 0.6791 0.1484 0.0352 0.0000 0.0144 0.0000 0.0000 0.0000 0.0000

0.0244 0.6975 0.1650 0.0368 0.0170 0.0281 0.0054 0.0112 0.0045 0.0102

0.0628 0.3788 0.4536 0.0577 0.0322 0.0037 0.0000 0.0062 0.0050 0.0000

0.3843 0.4571 0.1002 0.0279 0.0111 0.0082 0.0042 0.0038 0.0011 0.0022

0.1604 0.6561 0.1260 0.0173 0.0144 0.0027 0.0037 0.0010 0.0014 0.0169

0.0395 0.7197 0.2094 0.0205 0.0017 0.0029 0.0000 0.0008 0.0000 0.0055

0.0668 0.4753 0.3741 0.0664 0.0068 0.0026 0.0026 0.0000 0.0000 0.0053

0.1412 0.2087 0.5490 0.0820 0.0191 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0000 0.2722 0.7028 0.0167 0.0083 0.0000 0.0000 0.0000 0.0000 0.0000

0.0817 0.1239 0.2076 0.3129 0.2061 0.0306 0.0373 0.0000 0.0000 0.0000

0.0000 0.0000 0.4762 0.3963 0.1037 0.0238 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0000 0.0667 0.2667 0.4000 0.2000 0.0333 0.0000 0.0333 0.0000

#0.0000 0.0000 0.9115 0.0847 0.0038 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0071 0.4561 0.4240 0.1017 0.0000 0.0000 0.0000 0.0000 0.0110

0.0000 0.0042 0.2221 0.6694 0.0427 0.0096 0.0255 0.0000 0.0085 0.0180

0.0000 0.2582 0.1525 0.5145 0.0488 0.0102 0.0000 0.0091 0.0068 0.0000

0.0000 0.0358 0.8958 0.0184 0.0125 0.0234 0.0105 0.0000 0.0035 0.0000

0.0000 0.0067 0.5564 0.3842 0.0048 0.0064 0.0118 0.0052 0.0017 0.0229

0.0245 0.3412 0.2788 0.0678 0.1711 0.0831 0.0335 0.0000 0.0000 0.0000

0.0116 0.0490 0.3028 0.2021 0.0592 0.2156 0.1031 0.0440 0.0076 0.0052

0.0741 0.4870 0.1140 0.1056 0.0658 0.0145 0.0744 0.0441 0.0161 0.0045

#0.0257 0.3790 0.5195 0.0239 0.0353 0.0048 0.0086 0.0000 0.0000 0.0000 0.0000 0.0000 0.0032

#0.0000 0.7157 0.0928 0.0470 0.0718 0.0319 0.0408 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.1229 0.6791 0.1484 0.0352 0.0000 0.0144 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0244 0.6975 0.1650 0.0368 0.0170 0.0281 0.0054 0.0112 0.0045 0.0000 0.0000 0.0000 0.0102

#0.0628 0.3788 0.4536 0.0577 0.0322 0.0037 0.0000 0.0062 0.0050 0.0000 0.0000 0.0000 0.0000

#0.3843 0.4571 0.1002 0.0279 0.0111 0.0082 0.0042 0.0038 0.0011 0.0000 0.0000 0.0000 0.0022

#0.1604 0.6561 0.1260 0.0173 0.0144 0.0027 0.0037 0.0010 0.0014 0.0008 0.0033 0.0019 0.0108

#0.0395 0.7197 0.2094 0.0205 0.0017 0.0029 0.0000 0.0008 0.0000 0.0000 0.0000 0.0000 0.0055

#0.0668 0.4753 0.3741 0.0664 0.0068 0.0026 0.0026 0.0000 0.0000 0.0000 0.0000 0.0000 0.0053

#0.1412 0.2087 0.5490 0.0820 0.0191 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0000 0.2722 0.7028 0.0167 0.0083 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0817 0.1239 0.2076 0.3129 0.2061 0.0306 0.0373 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0000 0.4762 0.3963 0.1037 0.0238 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0000 0.0667 0.2667 0.4000 0.2000 0.0333 0.0000 0.0333 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0000 0.9115 0.0847 0.0038 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0071 0.4561 0.4240 0.1017 0.0000 0.0000 0.0000 0.0000 0.0110 0.0000 0.0000 0.0000

#0.0000 0.0042 0.2221 0.6694 0.0427 0.0096 0.0255 0.0000 0.0085 0.0000 0.0000 0.0000 0.0180

#0.0000 0.2582 0.1525 0.5145 0.0488 0.0102 0.0000 0.0091 0.0068 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0358 0.8958 0.0184 0.0125 0.0234 0.0105 0.0000 0.0035 0.0000 0.0000 0.0000 0.0000

#0.0000 0.0067 0.5564 0.3842 0.0048 0.0064 0.0118 0.0052 0.0017 0.0024 0.0062 0.0035 0.0108

#0.0245 0.3412 0.2788 0.0678 0.1711 0.0831 0.0335 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

#0.0116 0.0490 0.3028 0.2021 0.0592 0.2156 0.1031 0.0440 0.0076 0.0034 0.0017 0.0000 0.0000

#0.0741 0.4870 0.1140 0.1056 0.0658 0.0145 0.0744 0.0441 0.0161 0.0029 0.0000 0.0000 0.0016

#HB Discards

#HB Discards

#Headboat discard Index

#Starting and ending years of CPUE index (only <20" fish)

2005

2014

#Observed CPUE and CVs

0.56 0.41 2.02 1.39 0.63 0.56 0.41 2.02 1.39 0.63

0.30 0.37 0.17 0.21 0.27 0.30 0.37 0.17 0.21 0.27

#Starting and ending years of discards time series, respectively

1984

2014

2010 #year of the hook reg

#Observed discards (1000s) and assumed CVs

0.069 0.111 0.037 0.055 0.080 0.052 0.046 0.030 2.510 3.478 3.894 4.178 2.624
2.732 2.244 3.236 3.994 5.694 6.122 2.701 18.790 9.876 17.233 71.886 73.609
57.327 38.443 41.391 46.782 46.740 46.612

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

#Number and vector of years of headboat discard length composition data

10

2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

#sample sizes of length comps by year (first row number of trip, second row number of fish)

37.0 29.0 64.0 61.0 56.0 50.0 48.0 56.0 60.0 56.0

414.0 672.0 1499.0 1678.0 436.0 346.0 315.0 657.0 501.0 613.0

#HB discard length composition by year (year,lengthbin 3cm) all sizes

0.0021 0.0043 0.0193 0.0368 0.0549 0.1187 0.1806 0.1817 0.1686 0.1817 0.0450 0.0043 0.0000
0.0021 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0045 0.0164 0.0610 0.1458 0.2426 0.2351 0.1860 0.0595 0.0223 0.0208 0.0030 0.0015 0.0000
0.0015 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0090 0.0258 0.0314 0.0726 0.0982 0.1654 0.2209 0.2092 0.1143 0.0482 0.0032 0.0013 0.0006
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0044 0.0121 0.0214 0.0729 0.1621 0.1529 0.1581 0.1536 0.1376 0.1009 0.0221 0.0006 0.0006
0.0000 0.0006 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0046 0.0046 0.0302 0.0325 0.0838 0.1482 0.1448 0.1563 0.1914 0.1678 0.0333 0.0023 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

0.0000 0.0029 0.0231 0.0416 0.0375 0.0480 0.0667 0.0873 0.1000 0.1064 0.1006 0.0953 0.0757
0.0664 0.0589 0.0491 0.0058 0.0173 0.0029 0.0058 0.0000 0.0029 0.0000 0.0000 0.0000
0.0029 0.0029

0.0000 0.0032 0.0486 0.1360 0.1133 0.1019 0.0598 0.0356 0.0356 0.0453 0.0389 0.0630 0.0648
0.0662 0.0712 0.0324 0.0259 0.0227 0.0065 0.0097 0.0065 0.0032 0.0032 0.0000 0.0065
0.0000 0.0000

0.0000 0.0000 0.0215 0.0953 0.1403 0.1174 0.1540 0.0941 0.0958 0.0389 0.0299 0.0200 0.0297
0.0154 0.0154 0.0200 0.0215 0.0292 0.0200 0.0200 0.0108 0.0031 0.0031 0.0031 0.0015
0.0000 0.0000

0.0000 0.0000 0.0062 0.0461 0.1175 0.1137 0.1078 0.1196 0.1338 0.0866 0.0634 0.0371 0.0222
0.0103 0.0165 0.0124 0.0103 0.0173 0.0236 0.0268 0.0144 0.0144 0.0000 0.0000 0.0000
0.0000 0.0000

0.0181 0.1019 0.1709 0.1791 0.1429 0.0871 0.0631 0.0559 0.0335 0.0187 0.0246 0.0187 0.0181
0.0099 0.0105 0.0099 0.0066 0.0099 0.0089 0.0023 0.0033 0.0033 0.0016 0.0000 0.0016
0.0000 0.0000

########################General recreational (MRFSS/MRIP)##

#General Recreational landings

#Starting and ending years of landings time series, respectively

#1981

1955

2014

#Observed general recreational landings (1000s of fish) and assumed CVs

24.035 26.248 28.460 30.673 32.885 35.098 38.276 41.454 44.633 47.811 50.989 51.288 51.587
51.885 52.184 52.483 57.670 62.857 68.044 73.231 78.418 79.303 80.187 81.072 81.957
82.842 93.458 36.294 68.469 212.547 288.971 100.736 47.373 80.821 97.147 12.092 34.717
51.908 11.326 18.313 13.482 9.342 34.238 13.015 39.579 45.347 31.587 35.062 25.977
28.914 29.443 26.769 17.646 81.638 54.666 0.062 0.062 15.628 7.588 28.186

#93.458 36.294 68.469 212.547 288.971 100.736 47.373 80.821 97.147 12.092 34.717 51.908 11.326
18.313 13.482 9.342 34.238 13.015 39.579 45.347 31.587 35.062 25.977 28.914 29.443
26.769 17.646 81.638 54.666 0.062 0.062 15.628 7.588 28.186

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

#Gen rec Discards

#Starting and ending years of discards time series, respectively

1981

2014

2010 #year of hook reg

#Observed discards (1000s) and assumed CVs (average neighboring years to get 1982, 1986, & 1990)

4.435 4.435 4.435 61.825 64.088 64.088 64.088 50.274 19.383 19.383 19.383 27.994 68.149 66.540 50.890 20.445
16.574 26.789 162.710 248.597 202.665 123.362 159.329 199.638 72.855 119.735 288.276 511.984 240.516
138.478 33.484 142.961 83.992 285.962

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

#Number and vector of years of MRIP length composition data

#6

#1984 1985 1986 1987 1988 1989 #1999 2000 2001 2002 2003 2004 2005
2006 2007 2008 2009 2012 2013 2014

#sample sizes of length comps by year (first row number of trip, second row number of fish)

#12.0 15.0 82.0 16.0 27.0 15.0 #39.0 39.0 47.0 56.0 54.0 55.0 30.0
24.0 24.0 58.0 50.0 180.0 256.0 693.0

#67.0 39.0 225.0 69.0 82.0 48.0 #161.0 108.0 105.0 248.0 173.0 149.0 72.0
62.0 59.0 182.0 210.0 494.0 647.0 1917.0

#MRIP length composition by year (year,lengthbin 3cm)

#0.0597 0.1493 0.0896 0.2388 0.1642 0.0597 0.0000 0.0149 0.1642 0.0597 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0256 0.0256 0.1026 0.0256 0.1282 0.0513 0.0256 0.1282 0.1026 0.0000 0.1282 0.2051 0.0256
0.0256 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.7067 0.0711 0.1289 0.0533 0.0089 0.0133 0.0000 0.0044 0.0133 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0321 0.1124 0.0161 0.0293 0.1279 0.1866 0.1229 0.0720 0.0161 0.1357 0.0825 0.0399 0.0000
0.0133 0.0133 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.3591 0.0468 0.1249 0.0312 0.0063 0.0657 0.0468 0.0439 0.0595 0.0594 0.0125 0.0282 0.0344
0.0219 0.0468 0.0000 0.0063 0.0063 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0625 0.2500 0.1458 0.1042 0.0833 0.0625 0.0417 0.0417 0.0417 0.0208 0.0417 0.0000 0.0208
0.0417 0.0000 0.0417 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0405 0.2453 0.0913 0.1076 0.0568 0.0214 0.0082 0.0152 0.0621 0.0590 0.0731
0.0984 0.0723 0.0000 0.0082 0.0163 0.0082 0.0082 0.0082 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0168 0.0269 0.0195 0.0195 0.0000 0.0254 0.1942 0.1781 0.1882 0.0797
0.0516 0.0127 0.0127 0.0509 0.0161 0.0127 0.0389 0.0228 0.0202 0.0127 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0095 0.0000 0.0095 0.0095 0.0190 0.0952 0.2476 0.1810 0.1048
0.0857 0.0667 0.0571 0.0095 0.0000 0.0190 0.0190 0.0095 0.0095 0.0381 0.0095 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0050 0.0050 0.0089 0.1034 0.1840 0.1997 0.1407
0.0995 0.0601 0.0595 0.0272 0.0272 0.0078 0.0446 0.0156 0.0039 0.0039 0.0039 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0044 0.0584 0.0372 0.0496 0.0744 0.0124 0.0344 0.0922 0.1318 0.1023
0.0475 0.0651 0.0871 0.0643 0.0387 0.0264 0.0475 0.0088 0.0000 0.0000 0.0088 0.0044
0.0044 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0067 0.0134 0.0268 0.1678 0.1745 0.1678
0.0872 0.0604 0.0805 0.0537 0.0537 0.0201 0.0201 0.0268 0.0134 0.0067 0.0067 0.0134
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0278 0.0139 0.0000 0.0278 0.0139 0.1250 0.1667 0.1111
0.0972 0.0694 0.0000 0.0694 0.1389 0.0278 0.0278 0.0417 0.0139 0.0139 0.0000 0.0139
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.1129 0.2097 0.1774
0.0484 0.0323 0.0323 0.0484 0.0323 0.0806 0.0645 0.0161 0.0484 0.0161 0.0323 0.0161
0.0161 0.0161

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0169 0.0339 0.0000 0.0000 0.0169 0.2373 0.2034 0.0508
0.1017 0.0847 0.0508 0.0000 0.0339 0.0508 0.0169 0.0847 0.0000 0.0169 0.0000 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0055 0.0000 0.0055 0.0275 0.2088 0.2582 0.2198
0.1264 0.0440 0.0110 0.0110 0.0000 0.0220 0.0220 0.0055 0.0220 0.0000 0.0055 0.0000
0.0000 0.0055

#0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0238 0.1667 0.2000 0.1286
0.1190 0.1571 0.0905 0.0619 0.0143 0.0095 0.0143 0.0000 0.0000 0.0095 0.0048 0.0000
0.0000 0.0000

#0.0000 0.0000 0.0018 0.0165 0.0037 0.0165 0.0367 0.0330 0.0478 0.0416 0.0171 0.0275 0.0294
0.0404 0.0801 0.0753 0.1273 0.1236 0.1158 0.0710 0.0453 0.0245 0.0128 0.0043 0.0000
0.0061 0.0018

#0.0000 0.0015 0.0015 0.0046 0.0046 0.0263 0.0309 0.0325 0.0355 0.0417 0.0355 0.0402 0.0340
0.0417 0.0402 0.0417 0.0541 0.1020 0.1252 0.1298 0.0819 0.0448 0.0309 0.0124 0.0062
0.0000 0.0000

#0.0000 0.0000 0.0000 0.0016 0.0026 0.0104 0.0386 0.0490 0.0386 0.0292 0.0214 0.0177 0.0245
0.0417 0.0443 0.0449 0.0569 0.0668 0.1054 0.1560 0.1320 0.0709 0.0271 0.0136 0.0057
0.0010 0.0000

#Number and vector of years of MRIP age composition data

#11

10

#2001 2002 2003 2004 2005 2006 2007 2009 2012 2013 2014

2001 2002 2003 2004 2005 2006 2009 2012 2013 2014

#sample sizes of age comps by year (first row number of trip, second row number of fish)

15 84 91 83 78 26 58 121 139 315

43 262 354 312 338 169 463 1664 1467 3325

#MRIP age composition by year (year,age to 13)

0.0000 0.1841 0.7064 0.0848 0.0000 0.0247 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.1313 0.5610 0.1948 0.0538 0.0338 0.0111 0.0064 0.0000 0.0026 0.0000 0.0000 0.0052

0.0000 0.1034 0.3377 0.3520 0.1036 0.0320 0.0156 0.0183 0.0000 0.0199 0.0000 0.0000 0.0174

0.0000 0.1451 0.4051 0.2572 0.1094 0.0426 0.0085 0.0028 0.0032 0.0014 0.0013 0.0000 0.0233

0.0000 0.0000 0.3648 0.3518 0.1647 0.0841 0.0085 0.0097 0.0000 0.0056 0.0000 0.0026 0.0082

0.0000 0.0000 0.0168 0.4430 0.2365 0.1110 0.0548 0.0353 0.0278 0.0024 0.0024 0.0000 0.0700

#0.0000 0.0000 0.0000 0.5346 0.2895 0.1111 0.0602 0.0046 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0104 0.3308 0.6188 0.0064 0.0107 0.0068 0.0015 0.0000 0.0049 0.0098 0.0000 0.0000

0.0015 0.1264 0.1258 0.0583 0.3359 0.2119 0.0969 0.0077 0.0038 0.0094 0.0042 0.0041 0.0141

0.0105 0.0790 0.1512 0.1369 0.0381 0.2200 0.1703 0.1227 0.0035 0.0084 0.0114 0.0128 0.0351

0.0021 0.1339 0.0561 0.1308 0.1003 0.0307 0.1653 0.1828 0.1366 0.0032 0.0038 0.0147 0.0397

##########################Indices###

###MARMAP CVT index

#Starting and ending years of the index

2010

2014

#Observed index (units of fish/set) and assumed CVs

0.90 0.66 1.10 0.87 1.47

#0.66 0.66 1.10 0.87 1.47

0.26 0.23 0.18 0.20 0.17

#Number and vector of years of length composition data

#5

#2010 2011 2012 2013 2014

#sample sizes of length comps by year (first row number of trip, second row number of fish)

#74 70 155 143 151

#173 121 430 376 326

#length composition by year (year,lengthbin 3cm)

#0.0000 0.0058 0.0000 0.0058 0.0116 0.0174 0.0058 0.1098 0.1560 0.1040 0.1908 0.0693 0.0925
0.0636 0.0520 0.0347 0.0116 0.0405 0.0116 0.0058 0.0000 0.0000 0.0000 0.0116 0.0000
0.0000 0.0000

#0.0083 0.0166 0.0000 0.0166 0.0083 0.0083 0.0000 0.0083 0.0578 0.1073 0.0908 0.1073 0.1570
0.1239 0.0661 0.0661 0.0414 0.0249 0.0331 0.0331 0.0166 0.0000 0.0083 0.0000 0.0000
0.0000 0.0000

#0.0047 0.0257 0.0906 0.0443 0.0489 0.0860 0.1255 0.0720 0.0557 0.0209 0.0070 0.0093 0.0117
0.0303 0.0581 0.0443 0.0744 0.0580 0.0373 0.0349 0.0256 0.0070 0.0163 0.0070 0.0023
0.0000 0.0023

#0.0027 0.0240 0.0293 0.0506 0.0957 0.1303 0.0558 0.0744 0.0453 0.0586 0.0453 0.0239 0.0107
0.0186 0.0319 0.0054 0.0240 0.0611 0.0637 0.0612 0.0399 0.0160 0.0240 0.0080 0.0000
0.0000 0.0000

#0.0032 0.0272 0.0320 0.0304 0.0624 0.1117 0.1660 0.2140 0.0893 0.0544 0.0192 0.0032 0.0112
0.0128 0.0064 0.0096 0.0096 0.0176 0.0256 0.0480 0.0256 0.0128 0.0064 0.0016 0.0000
0.0000 0.0000

#Number and vector of years of MARMAP age composition data

5

2010 2011 2012 2013 2014

#sample sizes of age comps by year (first row number of trip, second row number of fish)

73 70 148 139 150

167 120 416 368 615

#age composition by year (year,age to 13+)

0.0060 0.1317 0.4850 0.2335 0.1138 0.0120 0.0000 0.0060 0.0000 0.0000 0.0120 0.0000 0.0000

0.0250 0.0583 0.3333 0.3833 0.1417 0.0417 0.0000 0.0000 0.0083 0.0000 0.0000 0.0000 0.0083

0.0505 0.3775 0.1538 0.0889 0.1875 0.0769 0.0288 0.0024 0.0072 0.0000 0.0000 0.0000 0.0264

0.2310 0.3125 0.1114 0.0435 0.1005 0.0924 0.0571 0.0163 0.0000 0.0027 0.0082 0.0054 0.0189

0.1642 0.4244 0.2244 0.0341 0.0163 0.0325 0.0407 0.0390 0.0130 0.0000 0.0016 0.0049 0.0049

###SERFS Video index

#Starting and ending years of the index

#2010

#2014

#Observed index (units of fish/set) and assumed CVs

#1.21 0.59 1.06 0.80 1.35

#0.22 0.17 0.14 0.12 0.14

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

```
###-- 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  #

##--------#-------#-------#-------#---------#---------#-------#

######Population                                    ###### Biological input
####################################

 911.36     600.0   1500.0  -4      911.36     -0.25      1    # VonBert Linf (units in mm TL)

  0.24      0.03      0.5   -3       0.24     -0.25      1    # VonBert K (units in mm TL)

 -0.33      -4.0      0.5   -4      -0.33     -0.25      1    # VonBert t0 (units in mm TL)

0.120227092434652         0.05       0.4         4           0.1         -0.25      3             # CV of length at age

######Fishery-dependent

   927     600.0   1500.0  -4         927     -0.25      1    # VonBert Linf (units in mm TL)

  0.22      0.03      0.5   -3       0.22     -0.25      1    # VonBert K (units in mm TL)

 -0.66      -4.0      0.5   -4      -0.66     -0.25      1    # VonBert t0 (units in mm TL)

0.109543670935091         0.05       0.4         4           0.1         -0.25      3             # CV of length at age

####Headboat

###  1014    600.0   1500.0    -4       1014     -0.25      3     # VonBert Linf (units in mm TL)

###  0.15      0.03      0.5   -3       0.15     -0.25      3     # VonBert K (units in mm TL)

### -1.35      -4.0      0.5    -4      -1.35     -0.25      3     # VonBert t0 (units in mm TL)

###   0.1      0.05      0.4    4        0.1     -0.25      3     # CV of length at age

######Rec

###  887      600.0   1500.0   -4        887     -0.25      3     # VonBert Linf (units in mm TL)
```

```
### 0.26 0.03 0.5 -3 0.26 -0.25 3 # VonBert K (units in mm TL)
### -0.26 -4.0 0.5 -4 -0.26 -0.25 3 # VonBert t0 (units in mm TL)
### 0.1 0.05 0.4 4 0.1 -0.25 3 # CV of length at age
######Commercial
### 902 600.0 1500.0 -4 902 -0.25 3 # VonBert Linf (units in mm TL)
### 0.24 0.03 0.5 -3 0.24 -0.25 3 # VonBert K (units in mm TL)
### -0.56 -4.0 0.5 -4 -0.56 -0.25 3 # VonBert t0 (units in mm TL)
### 0.1 0.05 0.4 4 0.1 -0.25 3 # CV of length at age
###20 in size limit
938 600.0 1500.0 -4 938 -0.25 1 # VonBert Linf (units in mm TL)
0.17 0.03 0.5 -3 0.17 -0.25 1 # VonBert K (units in mm TL)
-2.41 -4.0 0.5 -4 -2.41 -0.25 1 # VonBert t0 (units in mm TL)
0.101548787613865 0.05 0.4 4 0.1 -0.25 3 # CV of length at age
##Constant M
0.134 0.02 0.5 -3 0.134 0.015 1 # constant M (used only to compute MSST=(1-M)SSBmsy)
###### SR parameters ######################################
0.99 0.21 0.999 -4 0.99 0.0225 1 # SR steepness parameter (0.84 is mode from beta distn)
15.6616837044712 10 16 1 13 -0.5 1 # SR log_R0 parameter
0.0 -1.0 1.0 -3 0.0 -0.5 1 # SR recruitment autocorrelation (lag 1)
0.603906315341592 0.2 1.2 4 0.6 -0.25 3 # s.d. of recruitment in log
space
##### Selectivity parameters ######################
######Commercial
2.19810322975041 1.1 9 5 2 -0.5 3 #BLOCK 1: comm handline
age at 50% selectivity ***before 20in size reg
5.09630461237393 0.1 10 5 3.3 -0.5 3 # comm longline slope of
ascending limb
3.69092115958803 1.1 9 5 3 -0.5 3 #BLOCK 2: comm handline
age at 50% selectivity ***after 20in size reg
3.5719492035103 0.1 10 5 3.5 -0.5 3 # comm handline slope of
ascending limb
```

3.3949719235301 1.1 9 5 3 -0.5 3 #BLOCK 3: comm handline age at 50% selectivity ***no size regs, only mini season

2.60690350763616 0.1 10 5 2 -0.5 3 # comm handline slope of ascending limb

###Recreational

2.02419484560261 0 7 5 1.7 -0.25 3 #BLOCK 1: HB/rec A501

3.7430440415279 0 5 5 3.7 -0.25 3 #slope1 of ascending limb

3.83068768341327 1 7 5 4 -0.25 3 #L502

0.620728644026094 0.001 10 5 0.5 -0.25 3 #slope2 of descending limb

2.46134109804989 0 5 5 3.5 -0.25 3 #BLOCK 2: HB/rec A501 ***after 20in size reg

4.52722838197835 0 5 5 2.4 -0.25 3 #slope1 of ascending limb

2.82422317998484 1 7 5 3.5 -0.25 3 #L502

0.493005858650431 0.001 10 5 0.5 -0.25 3 #slope2 of descending limb

1.72292373318574 0 4 5 2 -0.25 3 #BLOCK 3: HB ONLY A501 ***no size regs, only mini season

2.80481184025412 0 5 5 3.5 -0.25 3 #slope1 of ascending limb

2.64117077533854 1 7 5 2.1 -0.25 3 #L502

0.507274418645538 0.001 10 5 0.5 -0.25 3 #slope2 of descending limb

3.44488690501079 0 5 5 3.25 -0.25 3 #BLOCK 2: GR A501 ***after 20in size reg

1.42300557400449 0 5 5 1.75 -0.25 3 #slope1 of ascending limb

2.94295697398949 1 7 5 4 -0.25 3 #L502

0.610757082881173 0.001 10 5 0.5 -0.25 3 #slope2 of descending limb

4.42994444040232 1 10 5 2 -0.25 3 #BLOCK 3: GR ONLY A501 ***no size regs, only mini season

1.84724593093153 0 5 5 1.5 -0.25 3 #slope1 of ascending limb

##CVT

```
1.62161955043674    0.5    5    5    1    -0.5    3    #CVT age at 50% selectivity ***independent survey
3.83860804440337    -2     10   5    2    -0.5    3    # CVT slope of ascending limb
###Commercial discards 20 in size limit - double logistic
0.839313264293596   0      3    6    1    -0.15   3    #BLOCK 2:Comm discards L501 *** 20in size reg
0.47373013390461    0      5    6    0.5  -0.15   3    #slope1 of ascending limb
2.34863210432231    1      4    6    2    -0.15   3    #L502
1.23884702748386    0.001  10   6    1    -0.15   3    #slope2 of descending limb
###Commercial discards during moratorium - logistic or dome-shaped
2.59862140748184    0.5    9    5    1.5  -0.5    3    #Block 3  A50  #peak
2.12628501842846    -3     10   5    1    -0.5    3    #slope   #top
###Headboat discards 20 in size limit - dome-shaped
0.874649380174233   0      2    6    1    -0.15   3    #BLOCK 2: HB/rec L501 *** 20in size reg
0.435582131519914   0      5    6    0.5  -0.15   3    #slope1 of ascending limb
1.33465095773572    0.5    3    6    2    -0.15   3    #L502
2.61843290545978    0.01   10   6    1.1  -0.15   3    #slope2 of descending limb
####Headboat discards during moratorium - dome-shaped    #BLOCK 3:  ***no size regs
1.33623581376742    0      4    6    1.5  -0.15   3    # A50                #peak
0.456402114439989   0      5    6    0.5  -0.15   3    #slope1 of ascending limb
4.1404247661517     1      7    6    4    -0.15   3    #L502
0.449084511635592   0.001  10   6    0.5  -0.15   3    #slope2 of descending limb
###### Index catchability parameters ########################
-9.17154119140469   -16    -4   3    -8.5   -0.5   1    # comm handline CPUE (log q)
-12.7788857211708   -16    -8   3    -13.5  -0.5   1    # HB CPUE (log q)
-16.1249161853339   -16    -8   3    -13.5  -0.5   1    # GR CPUE (log q)
-8.93692236137576   -16    -4   3    -8.5   -0.5   1    # MARMAP CVT CPUE (log q)
###### Fishing mortality parameters ##########################
0.0302739251602907  0.001  0.2  4    0.03   -0.25  3    # F used to initialize popn, distributed among fleets in proportion to their early Fs
```

-4.77657600864768 -10 1 1 -2 -0.5 1 # comm handlines log mean F

-4.3080426659435 -10 1 1 -4 -0.5 1 # HB log mean F

-3.51678340695798 -10 1 1 -3 -0.5 1 # GR log mean F

-4.88286853209138 -10 1 1 -4 -0.5 1 # comm handline discards log mean F

-4.38601652160287 -10 1 1 -4 -0.5 1 # HB discards log mean F

-3.50759123964235 -10 1 1 -4 -0.5 1 # GR discards log mean F
***actually headboat discards

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 # comm handline discard 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 (1950-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 (1955-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 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 (1981-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0

commercial handline discard F dev initial guesses (1992-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 (1984-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 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 (1981-2014)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 (1975-2013)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 (20-1)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0

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

##-- BAM DATA SECTION: likelihood weights section

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

##################Likelihood Component
Weighting##

1.0 #landings

1.0 #discards

0.98 #cH index

0.25 #HB index

0.28 #HB Disc CPUE

#0.49 #VID index

1.2 #CVT/VID index

0.15 #cH len comps

0.30 #cH disc len comps

#0.25 #HB len comps

0.30 #HB disc len comps

#0.25 #CVT len comps

#0.013 #MRIP len comps

0.15 #cH age comps

0.17 #HB age comps

3.5 #CVT age comps

0.10 #MRIP age comps

0.0 #log N.age.dev residuals (initial abundance)

1.0 #S-R residuals

0.0 #constraint on early recruitment deviations

1.0 #constraint on ending recruitment deviations

0.0 #penalty if F exceeds 3.0 (reduced by factor of 10 each phase, not applied in final phase of optimization) fULL F summed over fisheries

0.0 #weight on tuning F (penalty not applied in final phase of optimization)

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

##-- BAM DATA SECTION: miscellaneous stuff section

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

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

1.65E-8

2.99

#whole weight to gutted weight conversion

1.10

#time-invariant vector of % maturity-at-age for females (ages 1-20+)

0.43 0.73 0.91 0.97 0.990 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000

#Proportion female by age

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

#Number of batches by age

22 45 63 77 88 97 104 109 114 117 120 122 123 125 126 126 127 128 128 128

#Fecundity parameter a

3.012E-8

#Fecundity parameter b

4.775

#Fecundity scalar

1.0E8

#time of year (as fraction) for spawning: this follows what was done in S10, which didn't account for spawning seasonality

0.663

#age-dependent natural mortality at age (ages 1-20+) -- scaled Charnov

0.595 0.364 0.271 0.222 0.193 0.174 0.162 0.153 0.146 0.142 0.138 0.135 0.133
0.132 0.130 0.129 0.129 0.128 0.128 0.127

#Max observed age

51

#Discard mortality constants

0.48 #cH1

0.37 #HB1

0.37 #GR1

0.38 #cH2

0.285 #HB2

0.285 #GR2

#Scalar applied to historic rec landings. Used in MCB analysis, included here only to transfer to the rdat output.

1.0

#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

0.0

DD q switch: Integer value (choose estimation phase, negative value turns it off)

-1

#density dependent catchability exponent, value of zero is density independent, est range is (0.1,0.9)

0.0

#SE of density dependent catchability exponent (0.128 provides 95% CI in range 0.5)

0.128

#Age to begin counting D-D q (should be age near full exploitation)

5.0

#Random walk switch:Integer value (choose estimation phase, negative value turns it off)

-3

#Variance (sd^2) of fishery dependent random walk catchabilities (0.03 is near the sd=0.17 of Wilberg and Bence

0.03

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

0.35

#Year for tuning F

2000

#threshold sample sizes ntrips (>=)for length comps (set to 99999.0 if sel is fixed):

20.0 #comm

10.0 #comm discards

#20.0 #HB

20.0 #HB discards

#20.0 #CVT

#20.0 #MRIP

##threshold sample sizes ntrips (>=) for age comps (set to 99999.0 if sel is fixed)

10.0 #comm cH

10.0 #HB

10.0 #CVT

10.0 #MRIP

##Ageing error matrix (columns are true age 1-20+, rows are ages as read for age comps: columns should sum to one)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

##Landings growth curve? 1 is yes, 0 is no.

##0

999 #end of data file flag